THE UNIVERSITY *of* LIVERPOOL

Bacteria-inspired Algorithms and Their Applications to Power System Optimisation

Thesis submitted in accordance with the
requirements of the University of Liverpool
for the degree of Doctor of Philosophy

in

Electrical Engineering and Electronics

by

Mengshi Li, B.Sc.(Eng.), M.Sc.(Eng.)

July 2010

Bacteria-inspired Algorithms and Their Applications to Power System Optimisation

by

Mengshi Li

Acknowledgements

My deepest gratitude goes first to Professor Q. H. Wu, my supervisor, for his enthusiastic encouragement, kind support, and thorough guidance. He has helped me through all the stages of research and writing. Without his consistent and illuminating instruction, this research work could not have reached its present form.

Secondly, I would like to express my heartfelt gratitude to Professor J. R. Saunders, who led me into the field of biology. His expertise in the field guided me through the research work.

I also owe my sincere gratitude to all of the members of the Intelligence Engineering and Industrial Automation Research Group, especially Dr. W. J. Tang, Mr. J. Buse and Dr. T. Y. Ji, who gave me their support and time on my studies. My thanks also go to the Department of Electrical Engineering and Electronics at the University of Liverpool, for providing the research facilities that made it possible for me to conduct this research. Last but not least, my thanks goes to my beloved parents for their loving considerations and great confidence in me through these years.

Abstract

This thesis focuses on the development of Bacteria-inspired Algorithms (BIAs) and their applications to power system optimisation. The thesis first investigates the background of Evolutionary Algorithms (EAs) and BIAs and their differences, then describes in detail the relevant bacterial behaviours that have inspired this research.

Most EAs are in the form of meta-heuristic optimisation algorithms, which solves computational problems by processing the predefined procedural, based on the framework of a fixed population. However, this fixed-population framework is contradicted with real-world biological phenomena, as EAs are not only time-consuming as applications, but they also fail to reach the full potential of their searching capability.

In order to overcome these drawbacks, a varying population framework is introduced, initially presenting a novel bacteria-inspired optimisation algorithm; the Bacteria Foraging Algorithm with Varying Population (BFAVP). Developed from a more realistic model of bacteria foraging patterns, which incorporates the varying population framework and the underlying mechanisms of bacterial chemotaxis, metabolism, proliferation, elimination and quorum sensing, BFAVP has been evaluated on the basis of three sets of benchmark functions, including high-dimensional unimodal and multimodal functions and low-dimensional multimodal functions, with consideration of the differences between BFAVP and other EAs.

For practical applications, BFAVP is employed to solve three power system problems: economic dispatch, where BFAVP solves the Optimal Power Flow (OPF) problem and significantly reduces the computational time by separating

the whole power system into partitions; voltage control, where BFAVP optimises the location of Flexible AC Transmission System (FACTS) devices to minimise the power loss; and harmonic parameter estimation, where BFAVP estimates the phases and the fundamental frequency, alongside a least square method to estimate amplitudes.

With the aim of reducing the population size and computational complexity, a Paired-bacteria Optimiser (PBO) is developed in this research, which utilises only two individuals in each iteration. PBO combines both gradient and random searching strategies, thus making the algorithm suitable for solving multimodal functions with fast adaptive performance. Evaluated on a set of high-dimensional multimodal functions, the simulation results have shown that PBO has a faster convergence rate on most of these functions in comparison with other EAs.

The applicability of PBO to engineering is demonstrated by its application to economic dispatch with both dynamic and stochastic loads. With an outstanding convergence rate, PBO minimises the fuel cost rapidly in the dynamic environment, and effectively optimises the mean and standard deviation of fuel costs in the stochastic environment.

Contents

List of Figures

List of Tables

List of Abbreviations

EA	Evolutionary Algorithm
BIA	Bacteria-inspired Algorithm
BFAVP	Bacterial Foraging Algorithm with Varying Population
PBO	Paired-bacteria Optimiser
OPF	Optimal Power Flow
PSO	Particle Swarm Optimiser
FACTS	Flexible AC Transmission System
LP	Linear Programming
NLP	Non-linear Programming
ANN	Artificial Neural Network
GA	Genetic Algorithm
EP	Evolutionary Programming
ES	Evolutionary Strategy
GP	Genetic Programming
CPSO	Constriction factor approach PSO
CLPSO	Comprehensive Learning PSO
PSOPC	Particle Swarm Optimiser with Passive Congregation
UPSO	Unified PSO
FIPS	Fully Informed Particle Swarm
CLPSO	Comprehensive Learning PSO
ACO	Ant Colony Optimiser
TSP	Travelling Salesman Problem

AS-TSP	Ant System-TSP
GSO	Group Search Optimiser
BFA	Bacterial Foraging Algorithm
GABF	GA with Bacterial Foraging
ABFA	Adaptive BFA
DBFA	Dynamic BFA
DNA	Deoxyribonucleic acid
ATP	Adenosine triphosphate
ABFOA	Adaptive Bacterial Foraging Optimisation Algorithm
FEP	Fast EP
GAVaPS	GA with Varying Population Size
APGA	Adaptive Population sized GA
PRoFIGA	Population Resizing on Fitness Improvement GA
CMA-ES	Covariance Matrix Adaptation ES
GAOT	GA Optimisation Toolbox
PPO	Particle-Pair Optimiser
PSNR	peak signal-to-noise ratio
PGEP	Pseudo-gradient based EP
QP	Quadratic Programming
ISO	Independent System Operator
SVC	Static var compensator
TCSC	Thyristor controlled series capacitor
TCPST	Thyristor-controlled phase shifting transformer
UPFC	Unified power flow controller
SLP	System load parameter
DFT	Discrete Fourier Transform

Chapter 1

Introduction

Optimisation refers to choosing the best solutions from a set of available alternatives. Optimisation algorithms solve problems by seeking the solutions which have the minimal or maximal evaluation value of the objective function. In the discussion of this thesis, the problem is formulated by an objective function, and the optimisation aims to search for the minimal evaluation value of the function.

This chapter highlights the advantages of this study, introduces the basic concepts of Evolutionary Algorithms (EAs), explains the background of bacterial foraging behaviours, and summarises the contributions of this research work. The layout of the thesis and an auto-bibliography are given at the end of the chapter.

1.1 Motivation and Objectives

The gradient-based methods have been well studied over the past half a century and largely applied for solving a wide range of engineering and public service problems [8]. However these methods work based on gradient-based search mechanisms and their convergence is largely dependent on initial points of research. Therefore, they are not suitable for the optimisation functions which are non-differentiable and non-convex and contain many local optima.

Over the past few decades a great deal of research activities have been

carried out in the field of computational intelligence. In this field, various biologically inspired optimisation algorithms have emerged based on the studies of genetic evolution, animal behaviour, swarm intelligence, *etc.* Unlike gradient-based algorithms, EAs are population-based and aim to find global optimal solutions. They are suitable for resolving the complex optimisation problem where there exist many local optima and mixed function variables such as continuous, discrete, and logic variables.

However, EAs are notorious for their heavy and unpredictable computational loads. The potential for applications of these algorithms is limited to large-scale systems, such as power systems, telecommunication networks, traffic systems, and biological data processing, due to their high computational complexity [9][10]. One of the motivations of this research work is that most EAs are based on a fixed-population framework, which introduces unnecessary random search and function evaluations in the optimisation process. The redundant computation load is caused by a lack of knowledge of the relationship between population size and the dimensionality of the objective function, as well as the relationship between population size and the complexity of the optimisation problem. Meanwhile, some of the engineering problems can not be solved by the EAs with large population sizes [11].

In order to overcome the computational complexity and time consuming issues occurred for solving complex high-dimensional optimisation problems, this research will further explore the potential of the understanding of biological systems. The research on modelling of animal behaviours [12] and bacterial foraging patterns [13] have been undertaken in the Intelligence Engineering and Automation research group, The University of Liverpool. The investigations presented in this thesis focus on modelling further details of bacterial foraging behaviours and developing more efficient optimisation algorithms for applications to complex high-dimensional optimisation problems.

1.2 Biologically Inspired Optimisation Algorithms

The advent of rapid, reliable and cheap computing power over the past few decades has transformed many fields of science and engineering. The multidisciplinary field of optimisation is no exception [14]. First of all, with fast computers, researchers and engineers can apply classical optimisation methods, such as Linear Programming (LP) [15], Non-linear Programming (NLP) [16], and Convex Programming [17], to large-scale problems. In these algorithms, LP is an optimisation technique for linear objective functions, subject to linear equality and linear inequality constraints; NLP is able to solve the problems that have a nonlinear objective function or nonlinear constraints; and Convex Programming is designed to optimise the objective functions with a convex landscape. In addition, however, researchers have developed a large number of new optimisation algorithms that operate in a rather different way than the classical methods, and that allow scientists and engineers to solve optimisation problems where the classical methods are not applicable. Meanwhile, these novel optimisation algorithms do not need to be run a large number of times in multimodal problems, which makes their application efficient [18].

1.2.1 Natural computation

Since living creatures first appeared on Earth, nature has been doing a marvellous job of solving complex problems. Evolutionary pressure forced natural systems to come up with highly optimised and effective solutions to sustain life. Therefore, the adaptation of problem-solving approaches found in nature can be very helpful. Biomimetics aims to apply methods and systems found in nature to the study and design of engineering systems and advanced technology. With the advancement of computer science, researchers have taken this concept further by simulating natural processes to solve computational problems. This emerging field is referred to as natural computation [19].

According to [20], natural computation can be divided into three main

branches:

- Computation inspired by nature, also known as natural computation, which draws models from nature to develop problem-solving techniques for complex problems.
- The simulation and emulation of nature by means of computing, which aims at creating patterns, forming behaviours and organisms to mimic various natural phenomena by synthetic processes, and thus results in increasing the understanding of nature and insights about computer models.
- Computing with natural materials, which uses natural materials to perform computation, substituting and supplementing the current silicon-based computers.

This research falls into the first category, *i.e.*, the computing is inspired by nature, such as EAs and Artificial Neural Networks (ANNs) [21]. EAs involve techniques used to implement mechanisms that are inspired by evolutionary behaviour, and they have been widely applied to optimise scientific and engineering problems due to their simplicity and flexibility [22] [23]. The ANN is a mathematical model or computational model based on biological neural networks, which can be used to model complex relationships between inputs and outputs or to find patterns in data [24].

1.2.2 Evolutionary algorithms

EAs are a branch of computer science, which use an iterative process to search for a desired location in the solution space. To obtain the solution, the population is selected by a random process. The fitness value of the objective function for each individual in EAs is calculated orderly. Such processes are often inspired by mechanisms of genetic evolution and swarm intelligence.

Genetic evolution-based EAs

The basic concept of Genetic Algorithm (GA) is to simulate the processes in natural system of evolution, specifically those that follow the principles of survival of the fittest. GA refers to a particular class of EAs that uses techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover [25]. GA is implemented in a computer simulation in which a population of abstract representations of candidate solutions to an optimisation problem evolves towards better solutions [26].

A general GA has the following five operations: initialisation, selection, crossover, mutation and termination [25]. Traditionally, solutions are represented in binary value, but other encodings are also possible [27]. The evolution starts from a population of randomly generated individuals in the solution space and occurs in generations. Initially, the population is randomly selected from the entire range of possible solutions. Occasionally, the solutions are seeded in areas where optimal solutions are likely to be found. In each generation, the fitness of every individual in the population is evaluated. Individual solutions are selected through a fitness-based process, where solutions with a better fit are often more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Most functions are stochastic and designed so that a few solutions with poorer fits are selected. This helps GA to preserve population diversity and prevent premature convergence. The most well-studied selection methods include roulette wheel selection and tournament selection [28]. An intermediate population is selected from the existing population based on the fitness values to reproduce a new population. Then multiple individuals are stochastically selected from the current population, and crossover and mutation are applied to produce a new population. The new population is then used in the next generation of the algorithm. The algorithm either terminates when a maximum number of generations has been produced, or stops when a satisfactory fitness level for the population has been reached.

Evolutionary Programming (EP) is another conventional evolutionary based

EA [29]. It was first used by Fogel in 1960 as a way to use simulated evolution as a learning process aiming in order to generate artificial intelligence.

Similar to GA, EP is based on the parallel evolution of the population. Each individual represents a potential solution to the problem. According to survival of the fittest, the filial generation in EP is generated from the fittest parent generation by ranking these individuals. EP not only conducts a continuous crossover operation but also capitalises on the mutation operation in the evolution process. For each individual, the mutation varies in severity, which affects the behaviour of the individual [10]. Thus, EP focuses on optimising continuous functions, which are widely found in mathematical and engineering problems. The contemporary variant, Fast EP (FEP), uses a Cauchy mutation instead of Gaussian mutation. By introducing the Cauchy mutation, FEP is more likely to generate an offspring further away from its parent than Gaussian mutation due to its long flat tails. According to the experimental studies, the improvement enhances the global search ability of EP [9].

Genetic Programming (GP) is a type of EA based on the evolutionary progress developed by Koza to solve various complex optimisations and searching problems [30]. Unlike most EAs, GP can not only be applied to search for the optimal solution but also be used to search for mathematical functions to describe an unknown model. Trial solutions are represented in memory as tree structures. Trees can be easily evaluated in a recursive manner. Every tree node has an operator function, and every terminal node has an operand, making mathematical expressions easy to evolve and evaluate. In GP, crossover is applied on an individual by simply switching one of its nodes with another node from another individual in the population. With the tree-based representation, the crossover operates as a replacement for the whole branch. Mutation affects an individual in the population. It can replace a whole branch from the selected node, or it can replace only the node's information [31]. These two operators are applied to the chromosome in each generation.

Evolutionary Strategy (ES) was initially proposed in the 1960s by Rechenberg and Schwefel [32] [33]. The primary operators in ES are mutation and

selection. In each generation, a real valued vector of object variables is created by mutating the parent with an identical standard deviation to each objective variable. The fitness value of the child individual is compared with the value of its parent, and the one with the better value survives. This selection mechanism is identified as (1+1)-ES.

ES may converge to premature results during the optimisation process due to a lack of diversity. To overcome this drawback, a multi-member ES with a parent, (μ+1)-ES, was proposed by Recherche [34]. In this ES, μ parent individuals are recombined from one offspring, which is mutated from the worst parent individual. This operation has a similar effect to the crossover process in GA. There are other ES variants based on this improvement, such as (1+λ)-ES, where λ mutants are generated from the same parent; (μ,?)-ES, where all parents are selected in every generation with a varying population; $(\mu+\lambda)$-ES, where the best μ individuals are produced from the union of the parent and a surviving offspring; and (μ, λ)-ES, where only the best μ offspring individuals are used to form the next parent generation.

Swarm intelligence-base EAs

Particle Swarm Optimiser (PSO) is a stochastic optimisation technique developed by Eberhart and Kennedy [35], which is inspired by computer simulations of various interpretations of the movement of organisms in a flock of birds or a school of fishes. The population of PSO is called a *swarm* and each individual in the population of PSO is called a *particle*.

The i^{th} particle in the k^{th} iteration has a current position in an n-dimensional search space, $X_i^k \in (B_{\text{lo}}, B_{\text{up}})$, and a current velocity Vel_i^k, where B_{lo} and B_{up} indicate the lower and upper boundary of the search space. In each iteration of PSO, the swarm is updated according to the following equations:

$$Vel_i^{k+1} = \omega Vel_i^k + c_1 r_1 (P_i^k - X_i^k) + c_2 r_2 (P_{\text{g}} - X_i^k) \quad (1.2.1)$$

$$X_i^{k+1} = X_i^k + Vel_i^{k+1} \quad (1.2.2)$$

where ω is the inertia weight of the particle, which reduces the velocity of the

particle each iteration, c_1 and c_2 are the learning factors of the particle, P_i is the best previous position of the i^{th} particle, and P_g is the best position among all the particles in the swarm. Equation (1.2.1) is a high-efficient global searching method, which encourages the current individual to move towards the location that has the best fitness value.

An important variant of standard PSO is the Constriction factor approach PSO (CPSO), which was proposed by Clerc [36]. By reducing the inertia weight ω in each iteration, CPSO ensures the convergence stability, which leads to higher quality solutions compared to the standard PSO in unimodal functions. However, an over-decreased inertia weight also reduces the velocity of particle, *i.e.*, CPSO is trapped more easily at local optima in multimodal optimisation problems. Other variants of the PSO have also been developed in recent years, such as PSO with Passive Congregation (PSOPC) [37], Unified PSO (UPSO), Fully Informed Particle Swarm (FIPS) and most recently, Comprehensive Learning PSO (CLPSO) [38]. These algorithms have produced encouraging results in both benchmark testing and real-world applications.

Another approach based on swarm intelligence was inspired by ant colony behaviour. By modelling and simulating ant foraging behaviour, brood sorting, nest building, and self-assembling, an optimisation algorithm called Ant Colony Optimiser (ACO) was proposed [39]. In the real world, ants wander randomly, and after finding food return to their colony while depositing pheromone trails. If other ants find a pheromone path, they are more likely to discontinue their random travel and instead follow the trail, subsequently reinforcing it if they, too, eventually find food [40]. The pheromone trails evaporate with time, resulting in a reduction in the strength of attraction. This progress is simulated by ACO to find an optimal solution.

ACO has been applied to many combinatorial optimisation problems, which have discrete feasible solution spaces. The applications of ACO ranging from qua-dratic assignment to protein folding and vehicles routing. A number of derived methods have been adopted to solve dynamic problems, stochastic problems, multi-targets and parallel implementations [40]. In regards to the

Travelling Salesman Problem (TSP), the solution obtained by ACO is much better than the one found by a GA [41]. For the TSP with more than 50 cities, ACO [42] is able to minimise results in 2,500 iterations, with only 10 ants. After more than ten years of studies, both its application effectiveness and its theoretical background have been demonstrated, making the ACO a successful EA in the combinatorial optimisation domain [43].

Recently, a Group Search Optimiser (GSO) was developed at the University of Liverpool [44], which was inspired by animal searching behaviour and group living theory. Foraging behaviour may be described as an active movement by which an animal finds or attempts to find resources such as food, mates, oviposition or nesting sites, and it is perhaps the most prominent behaviour in which an animal engages [1]. To analyse the optimal policy for joining, two models have been proposed: information-sharing [45] and producer-scrounger models [46]. The framework of the GSO is primarily based on the PS model, which assumes group members search either for "finding" (producer) or for "joining" (scrounger) opportunities. Concepts from this framework are employed metaphorically to design optimum searching strategies for solving continuous optimisations. In GSO, some group members are also dispersed from their current positions to perform random walks, which discourages member from being trapped in local optima.

A group in GSO consists of three kinds of members: producers and scroungers, whose behaviours are based on the PS model, and dispersed members, who perform random-walk motions. At each iteration, a group member that is located in the most promising area and owns the best fitness value is chosen as the producer. Producer then stops and scans the environment to seek resources, as illustrated in Figure 1.1. Good scanning performance is essential for survival. These basic scanning strategies are introduced by white crappie, which is a genus of freshwater fish in the sunfish family [47]. The producer will scan at zero degrees and then scan laterally by randomly sampling two points in the scanning field. The scroungers will keep searching for opportunities to join the resources that are found by the producer. The rest of the group members will

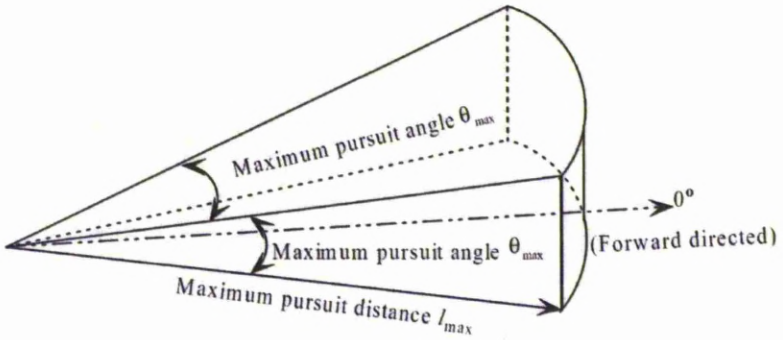


Figure 1.1: GSO scanning field in 3D space [1]

be dispersed from their current positions.

The GSO algorithm is better at solving multimodal functions, but has only modest performance when solving unimodal functions.

1.2.3 Bacterial foraging algorithm

A novel BIA based on bacterial behaviour was previously proposed, referred to as Bacterial Foraging Algorithm (BFA). The study of [48] elucidated the biology and physics behind the chemotactic behaviour of Escherichia coli (*E. coli*) bacteria, which live in human intestines. A variety of bacterial swarming and social foraging behaviours were further discussed. To implement BFA, a computer program that emulates the distributed optimisation process represented by the activity of social bacterial foraging was also presented in the research.

E. coli behaviours in BFA

Foraging theory in [48] is based on the assumption that bacteria search for and obtain nutrients in a way that maximises their energy intake per unit time spent foraging. Maximisation of such efficiency provides the nutrient sources necessary for survival as well as additional time for other activities.

Compared with other creatures, *E. coli* has a higher foraging efficiency due to its physiological structure. The flagella are tail-like projections that protrude from the cell body of a bacterium and function in locomotion. Bacterial flagella are motorised by a flow of H^+ ions or Na^+ ions. The motor of flagella filaments for foraging is quite efficient in that it uses only about 1,000 ions to make a complete revolution, and *E. coli* spends less than 1% of its energy on motility.

An *E. coli* bacterium can move in two different ways: it can tumble or it can swim. It alternates between these two modes of operation during its lifetime. If the flagella rotate clockwise, each flagellum pulls on the cell, and the net effect is that each flagellum operates relatively independently of the others, causing the bacterium to tumble. If the flagella move counterclockwise, their effects accumulate by forming a bundle: they essentially making a composite propeller and pushing the bacterium to move in one direction [49]. To tumble after a period of moving in one direction, the cell first slows down or stops. Because bacteria are such small creatures they experience almost no inertia only viscosity. When a bacterium stops swimming in the median, it stops within 1 μsecond [50].

The motion patterns that the bacteria generate in the presence of chemical attractants and repellents are called chemotaxis. The change in the concentration of a nutrient triggers a reaction that causes the bacterium to spend more time swimming and less time tumbling. As long as it travels up a positive concentration gradient, it will tend to lengthen the time it spends swimming, up to a certain point. The direction of movement is towards increasing nutrient gradients. The cell does not change its direction on a run due to the change in the gradient because the tumble determines the direction of the run. Typically, if the bacterium happens to swim down a concentration gradient, it will return to its baseline behaviour such that it is essentially searching for a way to climb back up the gradient.

Aside from chemotaxis behaviour, the sensory and decision-making system in *E. coli* has also been well studied in biology. The sensors in *E. coli* are the receptor proteins that are signalled to directly by external substances or via

the inducer, which is a type of protein for activating the gene expression. The sensor is extremely sensitive, in some cases requiring less than ten molecules of attractants to trigger a reaction; attractants can trigger a swimming reaction in less than 200 ms. Research [51] shows that the internal bacterial decision-making processes involve an integral feedback control mechanism.

However, it is possible that the local environment where a population of bacteria live can change either gradually or suddenly, due external influences. Events can occur such that all the bacteria in a region are killed or a group is dispersed into a new territory. The elimination has the possible effect of destroying chemotactic progress. However, it can also assist in chemotaxis because dispersal may place bacteria near nutrition sources. From a broad perspective, elimination and dispersal are part of the population-level long-distance motile behaviour. In summary, BFA employs the *E. coli* behaviours including chemotaxis, sensing, and elimination. A comprehensive discussion on bacterial behaviours will be further presented in Section 1.3.

Chemotaxis models

Based on these three behaviours, Passino [48] proposed a mathematical model to describe bacterial motility and swarming. In a n-dimensional searching space, suppose that the objective of the optimisation is to find the minimum of $F(X), X \in \mathbb{R}^n$, and the algorithm does not have measurements or an analytical description of the gradient $\nabla F(X)$. To use the bacteria to solve this optimisation problem, X is described as the position of a bacterium. $F(X)$ represents the combined effects of attractants and repellents from the environment, such that $F(X) < 0$, $F(X) = 0$, and $F(X) > 0$ implies that the bacterium at location X is in nutrient-rich, neutral, and noxious environments, respectively. Chemotaxis is a foraging behaviour that implements a type of optimisation, where bacteria try to climb the nutrient concentration ladder, *i.e.*, the bacterium aims to find the lowest value of $F(X)$. Meanwhile, the bacterium has to avoid noxious substances and search for ways out of neutral media. This process is implemented as a type of biased random walk.

Define a chemotactic step to be a tumble followed by a tumble, or a tumble followed by a run. Let j be the index for the chemotactic step. Let k be the index for the reproduction step. Let l be the index of the elimination-dispersal event. Then the position of each member in the population of S bacteria at the j^{th} chemotactic step, k^{th} reproduction step, and l^{th} elimination-dispersal event can be represented as $X_i^{j,k,l}, i = 1, 2, ..., S$. Let $F(X_i^{j,k,l})$ donates the cost at the location of the i^{th} bacterium $X_i^{j,k,l} \in \mathbb{R}^n$.

Let N_{c} be the length of the lifetime of the bacteria, as measured by the number of chemotaxis steps. Let $c_{\text{r}} > 0$ denote a basic chemotactic step, which is the length of the steps during runs. A tumble is represented by a unit length with random direction as ϕ^j. The tumble process is expressed as:

$$X_i^{j+1,k,l} = X_i^{j,k,l} + c_{\text{r}}\phi^j. \tag{1.2.3}$$

When the cost $F(X_i^{j+1,k,l})$ at $X_i^{j+1,k,l}$ is better than the cost at $X_i^{j,k,l}$, another step of size c_{r} in this same direction will be taken. The swim is continued as long as it continues to reduce the cost, but only up to a maximum number of steps, N_{s}. The cell will tend to keep moving if it is headed in the direction of increasingly favourable environments.

In BFA, cell-to-cell signalling via an attractant is also employed. The strength of the attractant between the i^{th} bacterium and the m^{th} bacterium is expressed as $J_{\text{cc}}(X_i^{j,k,l}, X_m^{j,k,l})$. Let d_{attract} be the depth of the attractant released by the cell and w_{attract} be a measure of the width of the attractant signal. The cell also repels a nearby cell in the sense that it consumes nearby nutrients. Let $h_{\text{repelling}}$ be the height of the repellent effect and $w_{\text{repelling}}$ be a measure of the width of the repellent. Then the cell-to-cell signalling for the i^{th} bacterium can be expressed as:

$$\begin{aligned} J_{\text{cc}_i} &= \sum_{m=1}^{S} J_{\text{cc}}\Big(X_i^{j,k,l}, X_m^{j,k,l}\Big) \\ &= \sum_{i=1}^{S}\Big(-d_{\text{attract}}\exp\Big(-w_{\text{attract}}\sum_{o=1}^{n}(x_{i,o}^{j,k,l} - x_{m,o}^{j,k,l})^2\Big)\Big) \\ &\quad + \sum_{i=1}^{S}\Big(-d_{\text{repelling}}\exp\Big(-w_{\text{repelling}}\sum_{o=1}^{n}(x_{i,o}^{j,k,l} - x_{m,o}^{j,k,l})^2\Big)\Big), \end{aligned} \tag{1.2.4}$$

where $x_{i,o}^{j,k,l}$ is the value on the o^{th} dimension of the i^{th} bacterium at position $X_i^{j,k,l}$. As each cell moves, its $J_{\text{cc}}(X_i^{j,k,l}, X_m^{j,k,l})$ function also moves. The J_{cc_i} function is time varying in that if many cells come close together, there will be a high amount of attractant and hence an increasing likelihood that other cells will move towards the group due to the movements of all cells. Then the i^{th} bacterium will hill-climb on:

$$F(X_i^{j,k,l}) + J_{\text{cc}_i}, \tag{1.2.5}$$

so that the cells will try to find nutrients, avoid noxious substances, and at the same time try to move towards other cells, but not too close to them.

After N_{c} chemotactic steps a reproduction step is taken. BFA supposes that only half of the bacteria have sufficient nutrients for survival. For reproduction, the population is sorted in the order of ascending accumulated cost. Then half of the bacteria die, and each bacterium in the healthier half splits into two bacteria, which are placed at the same location. Let N_{ed} be the number of elimination-dispersal events, where for each elimination-dispersal events, each bacterium in the population is subjected to elimination-dispersal with the probability p_{ed}.

Comparison with other EAs

There are several similar aspects between BFA and the other EAs. The nutrient concentration function in BFA describes the landscape of the objective function, which is the same as the fitness functions in most EAs. Bacteria reproduction in BFA and selection in GA provides these algorithms with the ability to produce filial generation. After reproduction, the children are present in the same concentration, whereas with crossover or bacteria splitting, children are located in a region near their parents on the fitness landscape.

Despite similarities, the algorithms are not equivalent. Each algorithm has its own distinguishing features. The key difference is that the convergence process of BFA does not only depends on the fitness function, but is also affected by nutrient concentration in equation (1.2.5). The fitness function represents the

likelihood of survival for given phenotypic characteristics. The nutrient concentration represents nutrient and noxious substance concentrations. In GA, crossover represents mating and the resulting differences in offspring, which is ignored in the BFA. Moreover, mutation represents gene mutation and the resulting phenotypical changes and rather than physical dispersal in a geographical area.

Recent advances in BFA

Since its introduction in 2001, BFA has been intensively studied by researchers around the world. The current research tends to improve the algorithm, adjust the parameters, combine it with other optimisation algorithms and apply it to real-world applications. Meanwhile, there are also some drawbacks in BFA, such as the low convergence efficiency in chemotaxis process and large computational complexity in nutrient concentration calculation. There was some research focusing on overcoming the drawbacks in BFA.

One of the developments stemming from the standard BFA is the mutation combined BFA, which was proposed by Biswas [52]. The paper has presented an improved variant of the BFA algorithm by combining the PSO-based mutation operator with bacterial chemotaxis. The scheme that is presented attempts to make a judicious use of the exploration and exploitation abilities of the search space and is therefore likely to avoid false and premature convergence in many cases. The mutation is expressed as:

$$Vel_i^{j+1,k,l} = \omega Vel_i^{j,k,l} + c_1(P_g^{j,k,l} - X_i^{j,k,l}) \tag{1.2.6}$$

$$\hat{X}_i^{i,j+1,k} = X_i^{j+1,k,l} + Vel_i^{j+1,k,l} \tag{1.2.7}$$

where $Vel_i^{j,k,l}$ indicates the velocity vector of the i^{th} bacterium at the j^{th} chemotaxis step, k^{th} reproduction, and l^{th} elimination-dispersal event, ω indicates the inertia of velocity, c_1 indicates the learning factor, P_g indicates the position with the global best nutrient value, and $\hat{X}_i^{i,j+1,k}$ indicates the position of the bacterium i^{th} after the mutation.

Replacing the cell-to-cell signalling in BFA with the mutation in PSO, the

computational complexity is markedly reduced. However, without repelling, the algorithm may be trapped in a local optimium.

Similar work has also been carried out in [53], which integrates the BFA with several functions in GA. Two GA operators, the mutation operation and the crossover operation, are adopted in the GA with Bacterial Foraging (GABF) algorithm to improve the convergence rate of BFA. The mutation operation is based on the continuous GA adopted in [54]. In GABF, the mutation is dynamic and generated based on the stage of the optimisation. Meanwhile, a modified simple crossover is used for the GABF algorithm. In GA, the crossover is used to vary the sequence of chromosomes from one generation to the next. By marking one or more points on both parents chromosomes, the strings among these points are swapped between the two parents chromosomes. The modified real-valued crossover in GABF is expressed as:

$$\breve{X}_i^{j,k,l} = \lambda X_i^{j,k,l} + (1-\lambda) X_m^{j,k,l} \tag{1.2.8}$$

$$\breve{X}_m^{j,k,l} = (1-\lambda) X_m^{j,k,l} + \lambda X_m^{j,k,l}, \tag{1.2.9}$$

where $X_i^{j,k,l}$ and $X_m^{j,k,l}$ are the parent individuals before crossover, $\breve{X}_i^{j,k,l}$ and $\breve{X}_m^{j,k,l}$ are the individuals after crossover, and λ is randomly generated in $[0,1]$.

The convergence speed of BFA is increased by engaging these two operators. In the early stage of BFA, most bacteria move randomly and slowly in the searching space, and the weak cell-to-cell signals delay the attraction. In GBFA, the mutation and attraction randomly place the filial generation. As a result, the searching performance in the early stage is improved. However, these operations also increase the time consumption of the optimisation, and the numerical results obtained by the GABF are not obviously improved.

Another drawback of BFA is the step size, c_r. Overlarge step sizes cause sightless searching and lead the bacteria to cross the optima for the remaining chemotactic steps. Undersized step sizes, in contrast, make BFA too slow to converge. Thus, a fixed c_r cannot meet the demands of different problems. To overcome this drawback, an Adaptive BFA (ABFA) is proposed in [55]. In the ABFA, the step size is generated through delta modulation [56] based on the error between the demand result and the current evaluation value.

Based on this control method, an ideal convergence speed is observed in ABFA. The ABFA was applied to optimise both the amplitude and phase of the weights of a linear array of antennas for maximum array factor at any desired direction and nulls in specific directions. The results have shown that the algorithm is not only simple to implement but also provides for a faster solution when the searching space is large.

The flexibility of BFA in dynamic environments has aroused recent interest. In the research of [57], the Dynamic BFA (DBFA) has the capability of tracking a changing optimum continually over time. A selection process is introduced, which is based on a new scheme to enable better adaptability in a changing environment. The basic improvement in DBFA is the ability to maintain a suitable diversity for a global search, while the local search ability is not degraded and changes in the environment are also considered. In the new selection scheme, the bacteria that have experienced more nutrient-rich areas are more likely to be selected as parents for the next generation. DBFA has been applied to optimise the fuel cost of a power system when the load changes over time. Encouraging results have been obtained [58].

1.3 Introduction to Bacterial Behaviour

Since bacteria were first observed by Antonie van Leeuwenhoek in 1676 [59][60][61], bacteriology has become an increasingly popular subject both for scientists and biologists [62]. The name "bacteria" was given to these microorganisms by Christian Gottfried Ehrenberg in 1838.

Bacteria are the most diverse and widespread prokaryote and are now divided among multiple kingdoms, and their behaviours are highly dependent on their unique structure. Typically, bacterial cells have diameters ranging from 1 to 5 μm and a variety of shapes, allowing them to live in a variety of different substances. The most crucial component of bacteria is the cell wall. The wall provides physical protection and prevents the cell from bursting in the environment. The complex cell wall contains a network of sugar polymers

cross-linked by short chemical substances, serving as a molecular membrane. The membrane not only encloses the entire bacterium but also anchors other molecules that extend from its surface. The movement of bacteria is driven by the flagella filaments located around the cell wall. Within the surface, cytosol is a semifluid substance, that houses the organelles. Chromosomes, carrying genes in the form of deoxyribonucleic acid (DNA), are surrounded by cytosol. To be a member of the prokaryote, DNA is concentrated in a region called the nucleoid and no membrane separates this region from the rest of cell [2].

Bacteria were chosen as a template for BIAs because the adaptability of bacteria is more flexible than in animals. During the past decades, various optimisation algorithms have been developed with inspiration arising from behaviours of the most typical bacterium, *E. coli*, due to its directional movement in a liquid environments [48]. *E. coli* is commonly found in the lower intestine of homoiothermal animals [63] and lives on a wide variety of substrates with different colony behaviours [64]. The mathematical models in this research are also derived from *E. coli* behaviours. The major bacterial behaviours involved in this thesis are motility, metabolism, reproduction, and communication.

1.3.1 Motility

The motility of bacteria includes being attracted or repelled by certain stimuli, known as chemotaxis [65]. Some species of bacteria, such as *E. coli*, can move at speeds exceeding 50 μm per second, which is an efficient way of foraging. The structure that enables bacteria to move, flagella filaments, are either scattered over the entire cell surface or concentrated at one or both ends of the cell. In an environment with well-distributed nutrition, flagellated bacteria move randomly. However, in a heterogeneous environment, bacteria exhibit chemotaxis, and move towards nutrition and away from toxins. Moreover, it has been demonstrated that solitary *E. coli* cells exhibit chemotaxis towards other members of their species, enabling the formation of colonies [66].

The motor of the bacteria flagellum is a basal apparatus embedded in the cell wall and plasma membrane. The energy for the motor movement is pro-

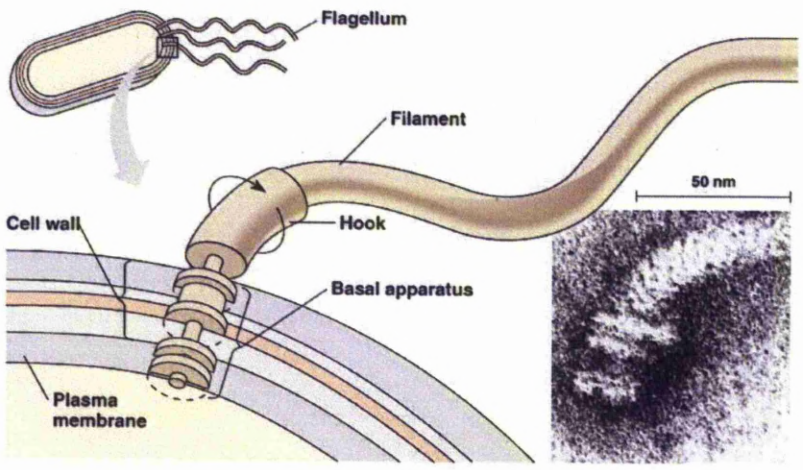


Figure 1.2: Structure of bacterial flagella [2]

vided by Adenosine triphosphate (ATP). The ATP-driven pumps transport ions out of the cell, and the diffusion of ions back into the cell powers the basal apparatus. This procedure drives a curved hook, and the hook is attached to a filament composed of chains of flagella. Flagella structure and its motor is illustrated in Figure 1.2.

E. coli has two distinct modes of movement: tumbling and running, which was discussed in Section 1.2.3. With the tumble-run process, bacteria are able to absorb nutrition from the environment. The substances and energy consumed by the tumble-run process are generated by their metabolic behaviour [67].

1.3.2 Metabolism

Bacterial metabolism is classified into nutritional groups on the basis of three major criteria: the energy used for growth, the source of carbon, and the electron donors used for growth [68]. *E. coli* metabolism is further divided into lithotrophs that use inorganic electron donors and organotrophs that use organic compounds as electron donors. *E. coli* uses the respective electron

donors for energy conservation and biosynthetic reactions. Respiratory organisms use chemical compounds as a source of energy by taking electrons from the reduced substrate and transferring them to a terminal electron acceptor in a redox reaction. This reaction releases energy that can be used to synthesise ATP. Meanwhile, sugars, amino acids, vitamins, and other nutrients are taken up by bacteria via specific transport systems localised in the cytoplasmic membrane.

The substances absorbed by bacteria are either stored inside the cell or consumed by metabolic processes. The substances provide the source for transmembrane ion potential, which is the major resource to retrieve the consumed flagella filaments in chemotaxis [69]. Meanwhile, ATP is synthesised during metabolism. However, the majority of the substances stored by the bacteria are used for reproduction.

1.3.3 Reproduction and elimination

The ability to proliferate in a fluid or within the cells of a living host is the key feature of bacteria that distinguishes them from commensal species. The primary objectives in this interaction are survival and multiplication [70]. Bacteria are unicellular organisms that increase population size via cell division. Each reproduction by cell division clones one daughter cell, which has the same DNA as the original parent cell. In the early stage of the bacteria life cycle, bacteria store resources and grow to a fixed size. Provided with sufficient resources, bacteria reproduce through binary fission, which is a form of asexual reproduction [71]. In a suitable environment, bacteria grow and divide extremely rapidly and populations can double as quickly as every 9.8 minutes [72].

Resources stored by *E. coli* metabolism are used as material for growth. In the bacterial lifespan, some organisms can grow extremely rapidly when nutrients become available. The growth of bacteria under laboratory conditions requires high levels of nutrients to produce large amounts of cells efficiently. However, limited nutrients exist in natural environments, which leads to the

survival of the fittest. As a result, bacterial evolution can be divided into several growth stages.

E. coli growth follows three stages. When a population of *E. coli* first enters a high-nutrient environment, the cells need to adapt to their new environment. The first phase of growth is the lag phase, a period of slow growth when the cells are adapting to the high-nutrient environment and are preparing for fast growth. The lag phase has high biosynthesis rates, as proteins necessary for rapid growth are produced [73]. The second phase of growth is the logarithmic phase, also known as the exponential phase. A rapid exponential growth is found in the logarithmic phase. During the logarithmic phase, nutrients are metabolised at the maximum speed until one of the nutrients is depleted and starts to limit growth. The final phase of growth is the stationary phase and is caused by depleted nutrients. The cells reduce their metabolic activity and consume cellular proteins. The stationary phase is a transition from rapid growth to a stress response state, and there is increased expression of genes involved in DNA repair, antioxidant metabolism and nutrient transport [74].

After the nutrition in the environment is consumed, the bacterial death rate is higher than the rate of bacterial growth. Bacteria run out of nutritional sources and begin the process of elimination.

1.3.4 Communication

Quorum sensing is defined as a type of decision-making process used by decentralised groups to coordinate behaviour. Many species of bacteria, such as *E. coli*, use quorum sensing to coordinate their gene expression according to the local density of their population [75].

Bacteria use quorum sensing to produce certain signalling molecules. These bacteria also have a receptor that can specifically detect the signalling molecule, which is called an inducer. When the inducer binds to the receptor, it activates the transcription of certain genes, including those for inducer synthesis. A bacterium detects the inducers from other bacteria in its environment, rather than from itself.

Quorum sensing enables bacteria to coordinate their behaviour. As environmental conditions often change very rapidly, bacteria need to respond quickly to survive. These responses include adaptation to the availability of nutrients, defence against other microorganisms, which may compete for the same nutrients, and avoidance of toxic compounds that are potentially dangerous for the bacteria. In *E. coli*, there are two types of inducers, which cause two different communication behaviours. The quorum sensing of *E. coli* either attracts the bacteria around the position, or repels the bacteria away from the group [76].

1.3.5 Summary

In conclusion, the bacterial behaviours adopted in this research are motility, metabolism, reproduction, elimination and communication. The major search principle of the proposed algorithms is developed from the *E. coli* motility model. Meanwhile, reproduction is integrated to enhance the searching performance. Last but not least, population diversity is maintained by elimination and communication.

1.4 Thesis Overview

This thesis is structured as follows:

- **Chapter 2** introduces a novel biologically inspired optimisation algorithm, Bacterial Foraging Algorithm with Varying Population (BFAVP), which is inspired from bacterial behaviours. Based on bacteria behaviours, such as chemotaxis, metabolism, proliferation, elimination and quorum sensing, BFAVP provides a varying population framework to solve complex problems. To evaluate this algorithm, a set of 13 benchmark functions is employed in the experimental studies, including 5 high-dimensional unimodal functions, 5 high-dimensional multimodal functions, and 3 low-dimensional multimodal functions. In this chapter, results of the benchmark functions obtained by GA, PSO, FEP, GSO, BFA, and Adap-

tive Bacterial Foraging Optimisation Algorithm (ABFOA) are also presented for sake of comparison. The comparison study shows that BFAVP has a superior performance compared to the other EAs for multimodal functions, in terms of accuracy and convergence speed. Bacterial colony behaviour is then discussed to further analyse the BFAVP. Meanwhile, colony behaviour in the BFAVP is compared with animal behaviours in other EAs.

- **Chapter 3** presents a Paired-bacteria Optimiser (PBO), which has only a pair of individuals in a population. In EAs, intensive computation caused by a large number of evaluations of the objective function required by all individuals reduces their performance in every single searching process. By studying single bacterium activity in media, a simplified optimisation algorithm with a small population size is proposed to solve complex optimisation problems. The proposed algorithm is tested on the 13 benchmark functions as described in Chapter 2. To demonstrate the advantages of PBO, the number of function evaluations in PBO is set to less than other EAs. According to the results of the comparison, it can be seen that PBO improves the search performance on the benchmark functions significantly and has a faster convergence speed and more accurate results than GA and PSO.

- **Chapter 4** describes four applications that are related to BFAVP. The first two applications are concerned with the Optimal Power Flow (OPF) problem. In the first application, BFAVP is applied to minimise the fuel cost of an IEEE 30-bus test system. The experimental results show that BFAVP reduces the fuel cost remarkably. However, the optimisation time is increased when the structure of the power system becomes more complex. To optimise the fuel cost in a complex power system, a distributed OPF solution is proposed, which is the second application. The power system is separated into several partitions, and each partition is optimised separately by BFAVP. According to the experimental results,

it can be concluded that BFAVP is suitable for solving the distributed OPF problems and outperforms both GA and PSO. To further reduce the real power loss of the power system, Flexible AC Transmission Systems (FACTS) devices are introduced. FACTS devices not only enhance the voltage profile but also reduce the real power loss. BFAVP is adopted to optimise the location of each device, due to the expensive cost of FACTS devices. Finally, BFAVP is also applied to estimate the parameters of the harmonics of power system signals.

- **Chapter 5** presents two applications of PBO. The real power demand of each bus is fixed in most OPF cases. However, in real-world power systems, the environment constantly changes with time. As a result, it is necessary to apply an adaptive optimisation algorithm. Compared to GA and PSO, the major advantage of PBO is its significantly few requirements in terms of the number of evaluations needed. To demonstrate this advantage, an experimental case is set up on a power system that has dynamic loads during the optimisation. From the experimental results, it can be seen that PBO can rapidly detect change in the environment, respond to the variation, and optimise the fuel cost. Then PBO is used in a stochastic optimisation case, which aims to reduce the mean and standard deviation of fuel cost in a power system with stochastic real power loads. The stochastic model of the power system is designed to meet the demand of real-world optimisation in an uncertain environment. The experimental results show that PBO is applicable to expensive objective function applications.

- **Chapter 6** concludes this thesis based on the experimental results obtained in this study. The merits and applications of BFAVP and PBO are further discussed. Additionally, this chapter includes suggestions for future work.

1.5 Auto-bibliography

The publications produced from this research work are listed in this section as follows:

1. W. J. Tang, **M. S. Li**, S. He, Q. H. Wu and J. R. Saunders, (2006), Optimal power flow with dynamic loads using bacterial foraging algorithm, *2006 International Conference on Power System Technology*, Vol. 6, pp. 1-5 Chongqing, China, September, 2006.

2. Z. Lu, **M. S. Li**, W. J. Tang and Q. H. Wu, (2007), Optimal location of FACTS devices by a bacterial swarming algorithm for reactive power planning, *2007 IEEE Congress on Evolutionary Computation*, pp. 2344-2349, Swissotel, The Stamford, Singapore, September, 2007.

3. **M. S. Li**, W. J. Tang, W. H. Tang, Q. H. Wu and J. R. Saunders, (2007), Bacterial foraging algorithm with varying population for optimal power flow, *In Lecture Notes in Computer Science* 4448, pp. 32-41, Mario Giacobini eds. Springer-Verlag. **(Best Paper Award)**

4. **M. S. Li**, W. J. Tang, W. H. Tang, Q. H. Wu and J. R. Saunders, (2007), Bacterial foraging algorithm with varying population for optimal power flow, *2007 European Workshop on the Application of Nature-inspired Techniques for Telecommunication Networks and other Parallel and Distributed Systems*, pp. 32-41, Valencia, Spain, September, 2007.

5. Z. Lu, **M. S. Li**, W. J. Tang and Q. H. Wu, (2007), Multi-objective optimisation of reactive power dispatch using a bacterial swarming algorithm, *2007 Chinese Control Conference*, pp. 460-464, Zhangjiajie, Hunan, China, July, 2007.

6. T. Y. Ji, **M. S. Li**, Z. Lu and Q. H. Wu, (2008), Optimal estimation of harmonics in dynamic environment using an adaptive bacterial swarming algorithm, *Submitted to IET Generation, Transmission & Distribution.*

7. Q. H. Wu, Z. Lu, **M. S. Li** and T.Y. Ji, (2008), Optimal placement of FACTS devices by a group search optimiser with multiple producer, *2008 IEEE Congress on Evolutionary Computation*, pp. 1033-1039, Hongkong, China, June, 2008.

8. **M. S. Li**, T. Y. Ji, Z. lu and Q. H. Wu, (2008), Optimal harmonic estimation using dynamic bacterial swarming algorithm, *2008 IEEE Congress on Evolutionary Computation*, pp. 1302-1308, Hongkong, China, June, 2008.

9. T. Y. Ji, **M. S. Li**, Z. Lu and Q. H. Wu, (2008) Optimal morphological filter design using a bacterial swarming algorithm, *2008 IEEE Congress on Evolutionary Computation*, pp. 452-458, Hongkong, China, June, 2008.

10. Z. Lu, **M. S. Li**, J. Lin and Q. H. Wu, (2008), Optimal allocation of FACTS devices with multiple objectives achieved by bacterial swarming algorithm, *2008 IEEE Power Engineering Society General Meeting*, pp. 1-6, Pittsburgh, Pennsylvania, USA, July, 2008.

11. Q. H. Wu, T. Y. Ji, **M. S. Li** and Z. Lu, (2008), Distributed optimal power flow using bacterial swarming algorithm, *2008 International Conference on Modelling, Identification and Control*, pp. 1-6, Shanghai, China, June, 2008.

12. W. J. Tang, **M. S. Li**, Q. H. Wu, and J. R. Saunders, (2008), Bacterial foraging algorithm for optimal power flow in dynamic environments, *IEEE Transactions on Circuits and SystemsI: Regular Papers*, Vol. 55, No. 8, pp. 2433-2442.

13. Q. H. Wu, T. Y. Ji, **M. S. Li** and Z. Lu, (2009), Distributed optimal power flow using bacterial swarming algorithm, *Accepted by International Journal of Modelling, Identification and Control.*

14. **M. S. Li**, W. J. Tang, T. Y. Ji, Q. H. Wu and J. R. Saunders, (2010), Bacterial foraging algorithms with varying population, *Accepted by BioSystems, BIO-D-10-00048.*

15. **M. S. Li**, T. Y. Ji and Q. H. Wu, (2010), Stochastic optimal power flow using paired bacteria optimiser, *Accepted by 2010 Power and Energy Society General Meeting.*

16. **M. S. Li**, T. Y. Ji, Q. H. Wu and J. R. Saunders, (2010), Optimal Dynamic Power Flow Using Paired-Particle Optimiser, *Prepared for a journal paper.*

1.6 Contributions of Research

Several contributions and outcomes made in this research are highlighted in this section.

- A novel optimisation algorithm, BFAVP, is developed in this research. Based on the studies of *E. coli* bacterial behaviours, a varying population size framework is introduced. Different from BFA, the mathematical models of the behaviours are enhanced to meet the demand of high-dimensional multimodal objective functions. The experimental results show that BFAVP has a superior performance in comparison with GA, PSO, FEP, and GSO.

- A PBO is developed, which aims to reduce the computation load of BFA. In PBO, a gradient-based local searching model is adopted. Then the algorithm improves the attraction model in PSO with a random repelling mechanism. By only employing two individuals in a population, the computational complexity of PBO is significantly reduced. The experimental results show that PBO outperforms other EAs on several benchmark functions with a greatly reduced requirement in terms of the number of evaluations. PBO is also applied to dynamic benchmark functions, demonstrating its suitability in dynamic environments.

- The proposed BFAVP has been applied to optimise the fuel cost in power systems. The OPF problem is investigated to obtain optimised operation conditions within specific constraints. Thus, the problem can be formulated as a constrained large-scale high-dimensional optimisation problem. The solution has previously been attempted by conventional gradient-based methodologies and, more recently, non-conventional methods, such as EAs [10]. Here, work has been undertaken on both IEEE 30-bus and IEEE 118-bus test systems. The experimental results show that BFAVP has superior performance over GA and PSO.

- To manage extremely complex power systems, it is necessary to separate the power system into small partitions. In partitioned power flow optimisation, the cost of the network can be optimised by coordinating the control of generators and taps in each subarea partition. A local reference bus is selected from the partition. The real and reactive power at both edges of the transmission lines between partitions is fixed to constant values. A complex case, the IEEE 118-bus test system, is employed to evaluate the performance of BFAVP for the distributed OPF problem. The experimental results show that BFAVP is able to optimise each partition simultaneously.

- FACTS devices are adopted in the simulation to reduce the real power loss in a power system. FACTS devices are revolutionising power transmission networks, resulting in the increasing efficiency and stability of power systems. However, due to the high cost of FACTS devices, it is important to place them optimally in a power network. Experimental studies are undertaken on the IEEE 30-bus test system. The fuel cost and real power loss of the system with the FACTS devices located by BFAVP are presented in the experimental studies.

- In electric power networks, harmonics exist in addition to the fundamental frequencies. These harmonics arise due to non-linear loads. Harmonic pollution significantly deteriorates the power quality. In some cases, the

fundamental frequency also deviates, and its deviation rate varies with time. BFAVP is applied to estimate the parameters of the harmonics in two environments with dynamic fundamental frequencies. The experimental results demonstrate that BFAVP is able to rapidly track the parameters, and this successfully improve the power quality.

- The work presented here also attempts to solve an OPF problem with the consideration of load changes alongside the optimisation process. A dynamic environment of a power system, which suffers from randomly occurring load changes on a number of buses simultaneously with a given probability, has been simulated. To trace the fast change in the environment and respond to the varying load effect, dynamic PBO is evaluated on two dynamic test systems, which are developed from the IEEE 30-bus test system and IEEE 118-bus test system. The experimental results obtained by evaluating PBO have been presented and discussed. The results demonstrate that PBO has great potential in its application to dynamic environments.

- In this research, a stochastic OPF model is adopted. Although stochastic OPF requires more computational loads than a static OPF, it is an accurate way when formulating a power system. Compared with other EAs, the computational complexity of PBO is significantly small, which makes it suitable for expensive objective functions. The experimental study demonstrates a less time-consuming solution for the stochastic OPF problem. It is found that PBO can effectively reduce the mean and standard deviation of the fuel cost in the stochastic power system environment with an outstanding convergence speed.

Part I

Developments of Bacteria-inspired Optimisation Algorithms

Chapter 2

Bacterial Foraging Algorithm with Varying Population

Nature-inspired optimisation algorithms [77], notably Evolutionary Algorithms (EAs), have been widely applied to solve various scientific and engineering problems. However, the performance of EAs is interfered by the random decisions in these algorithms [78]. Furthermore, due to the large population size, the huge number of evaluations in EAs causes more random decisions, and reduces efficiency of the algorithms in some applications. This chapter presents a novel nature-inspired heuristic optimisation algorithm: Bacterial Foraging Algorithm with Varying Population (BFAVP), based on a more bacterially realistic model of bacterial foraging patterns, which incorporates a varying population framework and the underlying mechanisms of bacterial chemotaxis, metabolism, proliferation, elimination, and quorum sensing. In order to evaluate its merits, BFAVP has been tested on several benchmark functions and the results show that it performs better than other popular EAs, in terms of both accuracy and convergence.

2.1 Introduction

In sociology and biology, a population is defined as a collection of interbreeding organisms of a particular species. Population size is an essential characteristic in evolution and has been incorporated in the development of EAs. Most EAs are based on a fixed population size, which introduces unnecessary computation in the optimisation process. The major drawback is the redundant computational load caused by lack of knowledge about 1) the relationship between population size and the dimensionality of the optimisation problem; and 2) the relationship between population size and the complexity of the optimisation problem. Some methods have been investigated that aim to give suggestions on choosing a proper population size for EAs; however, most of them focus on a specified EA [79], and cannot guarantee their predicted performance. Therefore, in most applications, the population size is blindly determined and the methods used are still based on a fixed population size.

To overcome this problem, a few EAs designed with a varying population size have been proposed over the past few years. The Genetic Algorithm with Varying Population Size (GAVaPS) [80] eliminates the population size as an explicit parameter by introducing the age and maximum lifetime properties for individuals. The maximum lifetimes are allocated depending on the fitness value of the new individual, while the age is increased at each generation by one. Individuals are removed from the population when their ages reach the value of their predefined maximal lifetime. This mechanism makes survivor selection unnecessary and population size an observable, rather than a parameter. The Adaptive Population sized GA (APGA) [81] employs a self-regulation method based on the distance among individuals to control the population size. The lifetime of the best individual in APGA remains unchanged when individuals grow older. The Population Resizing on Fitness Improvement GA (PRoFIGA) [82] resizes the population size based on improvements of the best fitness value in the population. On fitness improvement the algorithm becomes more biased towards exploration increasing the population size – short term lack of improvement makes the population smaller, but stagnation over a longer period

causes the population to grow again. As far as computational efficiency is concerned, however, these algorithms provide better performance than GAs that have a fixed population, and they are merely developed with some additional algorithmic improvements [83] [84]. Furthermore, these algorithms cannot be generalised for a wide range of optimisation problems, as their population size variation relies on the knowledge of specific optimisation problems. Last but not least, there is no evidence showing that these algorithms provide better optimisation results than GAs in generic cases. Meanwhile, the population size of some algorithms, such as Covariance Matrix Adaptation ES (CMA-ES), is determined by the dimension of the objective function [85].

The study of bacterial behaviours for development of novel optimisation algorithms has received a great deal of attention over the past few years. The Bacterial Foraging Algorithm (BFA) proposed by Kevin Passino is one of the emerging optimisation methods [48]. BFA stems from the study of *E. coli* chemotaxis behaviour and is claimed to have a satisfactory performance in optimisation problems. It has been applied to adaptive control systems [48], PID controller design [86], and dynamic environment optimisation [57]. However, Passino's BFA is also based on a fixed population size and demands a large amount of computation.

In this chapter, a novel optimisation algorithm, BFAVP, is proposed. Instead of simply describing chemotaxis behaviour in BFA, BFAVP involves further details of bacterial behaviours, and incorporates the mechanisms of metabolism, proliferation, elimination, and quorum sensing. This study builds on our previous study of a mathematical framework of bacterial foraging patterns in a varying environment [87]. In BFAVP, the following four features of bacterial behaviours are incorporated: 1) Chemotaxis, which offers the basic search principle of BFAVP. Chemotaxis comprises two basic foraging patterns, tumble and run. The biased random walk, which performs the "local search" in problem solving, is composed of the combination of these two patterns. In the tumble process, the heading angle of each bacterium is described as a compound angle, which is inspired by bacterial swimming behaviour in a 3-dimensional

space. 2) Metabolism, which controls the proliferation of a bacterium. In order to measure metabolism, the property of bacterial energy is incorporated, which is related to the nutrient level in the environment. Superior bacteria usually have more energy, which leads to a more filial generation. As a result, bacteria are able to aggregate around optima at an earlier stage of optimisation. 3) Proliferation and elimination, which result in a varying population. Proliferation occurs if bacterial energy is high enough. In order to provide a criterion for elimination, the property of bacterial age is incorporated in BFAVP. Bacterial age describes the life cycle of a bacterium. A bacterium with a higher age is prone to be eliminated. By this mechanism, with the same expectation value, the computational complexity of the optimisation process can be reduced and unnecessary computation can be avoided, by eliminating the bacteria that do not possess sufficient energy during their evolutionary process. 4) Quorum sensing, which enables BFAVP to escape from local optima. This is a two-fold operation that can either attracts a bacterium to the optima or repels it away from the location where bacteria concentrated.

In previous work, BFAVP has been applied to solve OPF problems and produced a superior result [88]. BFAVP has been evaluated on 13 benchmark functions, which are high-dimensional unimodal functions, high-dimensional multimodal functions, and low-dimensional multimodal functions respectively. The experimental results show the merits of the proposed algorithm in comparison with Genetic Algorithm (GA), Particle Swarm Optimiser (PSO), Fast Evolutionary Algorithm (FEP) [9], and Group Search Optimiser (GSO), respectively.

The rest of the chapter is organised as follows. Section 2.2 introduces the details of bacterial behaviours and mathematical models of BFAVP. The implementation of the algorithm is also given in this section. In Section 2.3, the experimental studies of the proposed BFAVP are presented with descriptions of the benchmark functions, experimental setting including the parameter setting of the BFAVP algorithm and the experimental results. The performance improvement based on bacterial colony behaviour in BFAVP is discussed in

Section 2.4, followed by the conclusion of the chapter in Section 2.5.

2.2 The BFAVP Algorithm

In order to describe the features of chemotaxis, metabolism, proliferation and elimination, and quorum sensing, four mathematical models were constructed, which will be elucidated in detail in this section. During an optimisation process, the four models were performed in order in each iteration, *i.e.*, an iteration comprised four procedures.

2.2.1 Chemotaxis

E.coli sense simple chemicals in the environment and are able to decide whether chemical gradients are directionally favourable or unfavourable [89]. If a favourable gradient is sensed, they will swim in that the direction. Bacteria swim by rotating thin, helical filaments known as flagella driven by a reversible motor embedded in the cell wall. *E.coli* has 8∼10 flagella placed randomly on the cell body [90]. In chemotaxis, the motor runs either clockwise or counterclockwise with the different directions of ions flowing through the cytoderm. When the motors turn clockwise, the flagellar filaments work independently, which leads to erratic displacement. This behaviour is called "tumble". When the motors turn counterclockwise, the filaments rotate in the same direction, thus pushing the bacterium steadily forward. This behaviour is called "run". The alternation of tumble and run is presented as a biased random walk.

In BFA, the heading angle of a bacterium is represented by a unit length with random direction. However, with the increase of dimension, the differences of heading angle between each iteration increases obviously. As a result, it is not able to describe an accurate direction for the bacterium to move. A method which extends the animal searching behaviour from 3-dimensional real-world environment to n-dimensional searching space is proposed in [44]. In this method, the heading angle is represented by a $(n-1)$-dimensional angle in the range of $[0, 2\pi]$. Then the direction of the search can be calculated

through polar-to-cartesian transform. By limiting the variation of the heading angle, the variation of searching direction is also restricted. It not only is a biologically-realistic searching model, but also improves the efficiency of the search.

The chemotaxis behaviour can be modelled by a tumble-run process that consists of a tumble step and several run steps. The tumble-run process follows a gradient searching principle, which indicates that the position of the bacterium is updated in the run steps by the gradient information provided by the tumble step. Determining the rotation angle taken by a tumble action in an n-dimensional search space can be described as follows. Suppose the p^{th} bacterium, in the tumble-run process of the k^{th} iteration, has a current position $X_p^k \in \mathbb{R}^n$, the objective of the optimisation is to find the minimum of $F(X_p^k)$. The bacterium also has a rotation angle $\varphi_p^k = \left(\varphi_{p1}^k, \varphi_{p2}^k, ..., \varphi_{p(n-1)}^k\right) \in \mathbb{R}^{n-1}$ and a tumble length $D_p^k(\varphi_p^k) = \left(d_{p1}^k, d_{p2}^k, ..., d_{pn}^k\right) \in \mathbb{R}^n$, which can be calculated from φ_p^k via a polar-to-cartesian coordinate transform:

$$\begin{aligned} d_{p1}^k &= \prod_{i=1}^{n-1} \cos\left(\varphi_{pi}^k\right), \\ d_{pj}^k &= \sin\left(\varphi_{p(j-1)}^k\right) \prod_{i=p}^{n-1} \cos\left(\varphi_{pi}^k\right) \quad j = 2, 3, ..., n-1, \\ d_{pn}^k &= \sin\left(\varphi_{p(n-1)}^k\right). \end{aligned} \tag{2.2.1}$$

In polar-to-cartesian coordinate transform, an arbitrary vector in the n-dimensional space can be represented by $n-1$ angles and a normalised distance to the original point.

The maximal rotation angle $\varphi_{\max}$ is related to the number of the dimensions of the objective function, which can be formulated as:

$$\varphi_{\max} = \frac{\pi}{\lfloor \sqrt{n+1} \rfloor}, \tag{2.2.2}$$

where n is the number of dimensions, and $\lfloor \cdot \rfloor$ denotes the operation which rounds the element to the nearest integer towards minus infinity. By introducing equation (2.2.2), the maximal rotation angle is restricted with the increase

of dimensionality. As a result, the algorithm is easier to converge to the optima in high-dimensional environment, when it finds a heading angle with an effective direction.

In the tumble-run process of the k^{th} iteration, the p^{th} bacterium generates a random rotation angle, which falls in the range of $[0, \varphi_{\max}]$. A tumble action takes place in an angle expressed as:

$$\hat{\varphi}_p^k = \varphi_p^k + r_1 \varphi_{\max}/2, \tag{2.2.3}$$

where $r_1 \in \mathbb{R}^{n-1}$ is a uniform random sequence with a range of [-1,1]. The run action immediately following the tumble action. Because the run action will be performed more than once, the position X_p^k is recorded as $\hat{X}_p^{k,0}$, which indicates the position of the p^{th} bacterium at the beginning of the k^{th} iteration.

Once the angle is determined by the tumble step, the bacterium will run for a maximum of N_{c} chemotaxis steps. If at the N_{f} $(N_{\text{f}} < N_{\text{c}})$ run step, the bacterium cannot find a position which has a higher fitness value than the current one, the run process also stops. The position of the p^{th} bacterium is updated at the h^{th} $(h \geq 1)$ run step in the following way:

$$\hat{X}_p^{k,h} = \hat{X}_p^{k,h-1} + r_2 D_p^k(\hat{\varphi}_p^k), \tag{2.2.4}$$

where $r_2 \in \mathbb{R}^1$ is a normally distributed random number generated from $\mathcal{N}(0, D_{\max})$, $D_{\max}$ is the maximal step length of a run, and $\hat{X}_p^{k,h}$ is the position of the p^{th} bacterium after the h^{th} run step. For convenience of description, the position of the p^{th} bacterium beginning immediately after the tumble-run process of the k^{th} iteration is denoted by $\hat{X}_p^{k,N_{\text{f}}}$, $N_{\text{f}} \leq N_{\text{c}}$.

The rotation angle is updated after each iteration. The tumble angle of the p^{th} bacterium at the beginning of the $(k+1)^{\text{th}}$ iteration is expressed as φ_p^{k+1}, which has the same value as $\hat{\varphi}_p^k$.

2.2.2 Metabolism

Heterotrophic bacteria are not able to produce energy by photosynthesis. A heterotrophic bacterium (such as *E. coli*) acquires energy by absorbing nutrients from the environment and uses them for chemotaxis, growth and sensing.

The objective of a bacterium is to survive in a nourishing environment and to reproduce. Provided with sufficient nutrition and energy, the bacterium first replicates its genetic material and then divides. However, in a hostile environment, some bacteria switch from germination to the sporulation cycle. Without sufficient energy, the bacterium stops metabolising, and its genetic material is packed into a small spore.

In BFAVP, bacterial energy is described as e_p^k, which is a measure of the energy quantity of the p^{th} bacterium at the k^{th} iteration. The fitness value of this bacterium at the k^{th} iteration, $F(X_p^k)$, is the source of its energy e_p^k. In each iteration, bacteria absorb energy subsequent to the tumble-run process. The energy transform of the p^{th} bacterium in the k^{th} iteration is defined as:

$$\tilde{e}_p^k = e_p^k + \alpha(F(X_p^k) - F_{\min}), \tag{2.2.5}$$

where α is a coefficient for energy transform and $F_{\min}$ indicates the minimal fitness value obtained in the history. For the p^{th} bacterium, if its energy $\tilde{e}_p^k$ is not high enough to induce proliferation, it will remain the same value until the next tumble-run process in the $(k+1)^{\text{th}}$ iteration, *i.e.*, $e_p^{k+1} = \tilde{e}_p^k$. On the other hand, if $\tilde{e}_p^k$ reaches the threshold, the proliferation process starts, which is explained in the following section.

2.2.3 Proliferation and elimination

Bacteria absorb energy from the nourishment environment, and perform a proliferation process if enough energy is gained. Bacteria grow to a fixed size and then reproduce through binary fission, which results in cell division. Two identical daughter cells are produced by cell division. Bacteria may also be eliminated if they reach the limitation of their lifespan by mimicking cell death.

Bacterial growth follows three phases. The lag phase is a period of slow growth during which bacteria acclimatise to the nutrient environment. Once the metabolic machinery is running, bacteria begin to multiply exponentially. The log phase is the period during which nutrients are consumed at the maxi-

mum speed. In the stationary phase, as more and more bacteria are competing for dwindling nutrients, growth stops and the number of bacteria stabilises [73].

In BFAVP, the proliferation process is controlled by the level of bacterial energy $\tilde{e}_p^k$. As soon as $\tilde{e}_p^k$ reaches the upper limit of the energy a bacterium can possess, the p^{th} bacterium turns into the proliferation state, which produces a new individual. For the new bacterium, bacterial energy is represented as:

$$\hat{e}_{S+1}^k = \tilde{e}_p^k/2, \tag{2.2.6}$$

where S is the population size in the current iteration, and the previous bacterium also keeps half of the energy as follows:

$$\hat{e}_p^k = \tilde{e}_p^k/2. \tag{2.2.7}$$

The new bacterium will not be involved in the optimisation process until the next iteration, so its energy may as well be denoted by e_{S+1}^{k+1}. For the p^{th} bacterium, its energy $\hat{e}_p^k$ will not change until the next metabolic charge takes place. In the application, the upper limit of the energy a bacterium can possess is set to be 50 times as its first fitness value.

Whether elimination behaviour occurs depends greatly on bacterial age. For the p^{th} bacterium at the k^{th} iteration, its age is recorded by an age counter, O_p^k. In the next iteration, the bacterial age becomes:

$$O_p^{k+1} = O_p^k + 1. \tag{2.2.8}$$

If a cell divides, the ages of the two daughter cells are set to 0. For a bacterium, when its age is close to the upper limit of its lifespan, it may be removed from the search space. However, assigning a fixed lifespan to each bacterium results in a large number of bacteria being eliminated at the same time. This is not in line with the real-world bacterial phenomenon. Therefore, in order to consider the randomness of elimination behaviour, a probability is introduced to describe the uncertainty of bacterial ages. The probability density of bacterial lifespan expectancy is set as:

$$\mathcal{P} = \frac{1}{\sqrt{2\pi}\sigma_{\text{span}}} \exp\left(-\frac{(O_p^{k+1} - \mu_{\text{span}})^2}{2{\sigma_{\text{span}}}^2}\right), \tag{2.2.9}$$

where μ_{span} is the mean of the bacterial lifespan, and σ_{span} indicates the standard deviation of all lifespan expectancies. Equation (2.2.9) illustrates that the lifespan follows a Gaussian distribution, which is in accordance with the biological behaviour in the real world. After the dead bacteria are eliminated, their positions are distributed around local optima, *i.e.*, BFAVP can detect all local optima in a single run. This particular advantage is explained in Section 2.4.

2.2.4 Quorum sensing

A bacterium uses a batch of receptors to sense the signals coming from external substances. The bacterium also has an inducer, which is a molecule inside the bacterium, to activate gene expression [48]. When the inducer binds to the receptor, it activates the transcription of certain genes, including those responsible for inducer synthesis. This process, called "quorum sensing", was discovered by Miller to explain cell-to-cell communication [91].

In BFAVP, most "nutrients" are located around optima, which correspond to higher fitness values. Based on this assumption, the density of the inducer is increased if the fitness value is greater. On the one hand, quorum sensing promotes symbiotic behaviour inside a single bacterial species, which attracts the bacteria to the global best position. Within a population, a certain percentage of bacteria are moving by attraction. Hence, inspired by PSO, the positions of the bacteria moving by attraction are updated as follows:

$$X_p^{k+1} = \hat{X}_p^{k,N_f} + r_3(X_{\text{best}} - \hat{X}_p^{k,N_f}), \tag{2.2.10}$$

where $r_3 \in \mathbb{R}^1$ is a normally distributed random number with a range of $[-1, 1]$, which describes the strength of bacterial attraction, and X_{best} indicates the position of the current best global solution updated after the evaluation of each function.

On the other hand, quorum sensing can occur between disparate species, which may cause competition or even be inimical between each species. In BFAVP, a small number of the bacteria are randomly selected to be repelled.

To measure the degree of repelling, a repelling rate is defined by ζ, *i.e.*, in each iteration, *e.g.*, 100ζ percent of the bacteria are processed by repelling. Accordingly the attraction rate is $100(1-\zeta)$ percent. The repelling process is based on the random searching principle. If the p^{th} bacterium shifts into the repelling process, a random angle in the range of $[0, \pi]$ is generated. The bacterium is thereby "moved" to a random position following this angle in the search space, which can be described as:

$$X_p^{k+1} = \hat{X}_p^{k,N_t} + r_4 D_p^k (\hat{\varphi}_p^k + \pi/2), \tag{2.2.11}$$

where $r_4 \in \mathbb{R}^n$ is a normally distributed random sequence which drawn from $\mathcal{N}(0, D_{\text{range}})$, and D_{range} is the range of the search space.

The pseudo code of BFAVP is listed in Table 2.1, and the flow chart is illustrated in Figures 2.1, 2.2 and 2.3. The notations used in BFAVP is listed in Appendix B.1.

Set $k := 0$;
Randomly initialise bacterial positions;
WHILE (termination conditions are not met)
FOR (each bacterium p)
Tumble: Turn the heading angle randomly within half of the $\varphi_{\max}$ by (2.2.3).
Set $h := 1$
Run:
WHILE ($h < N_c$)
Move the bacterium towards the heading angle to the new position $\hat{X}_p^{k,h}$ by equation (2.2.4). If the fitness value at current position is worse than that at previous position, set $h := N_c$; otherwise, increase h by 1;
END WHILE
END FOR
FOR (each bacterium p)
Proliferation:
Calculate $\tilde{e}_p^k$ by equation (2.2.5);
IF ($\tilde{e}_p^k$ reaches the upper limit of the energy a bacterium can possess)
Bacterium p divides into 2 daughter cells, which stay at the same position as the parent cell. Each new cell takes half of the energy of the original one, according to equations (2.2.6) and (2.2.7), both of the new cells' ages are set to 0;
END IF
Elimination:
Calculate bacterial age O_p^k by equation (2.2.8);
Generate a random lifespan by equation (2.2.9);
IF (O_p^k exceeds the lifespan)
Bacterium p is eliminated. Record its position for locating local optima;
END IF
END FOR
Quorum Sensing:
(1-ζ)100% of the bacteria are attracted to the global optimum by equation (2.2.10), ζ100% of bacteria are repelled by equation (2.2.11);
$k := k + 1$;
END WHILE

Table 2.1: Pseudo code of BFAVP

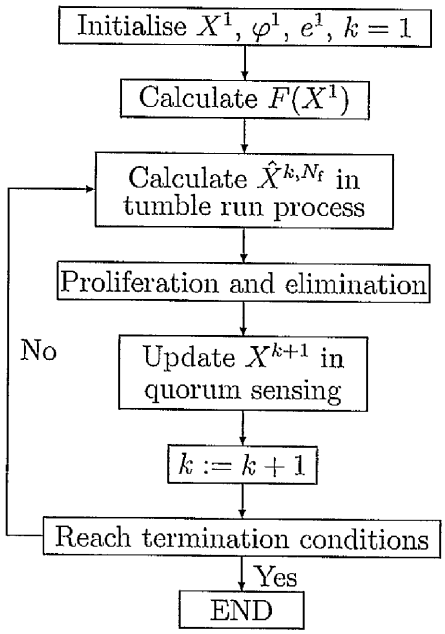


Figure 2.1: The main flow chart of BFAVP

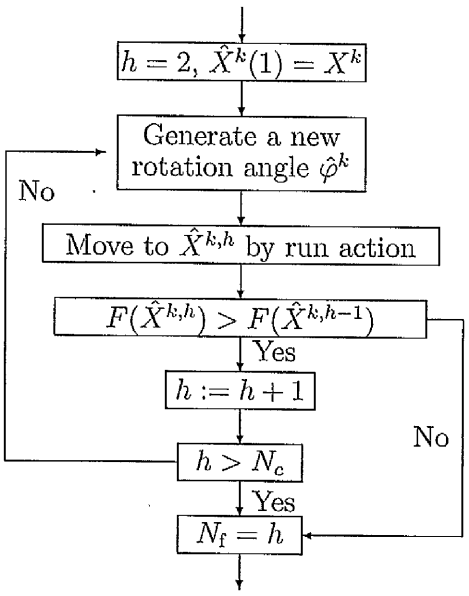


Figure 2.2: The flow chart of tumble run processes in BFAVP

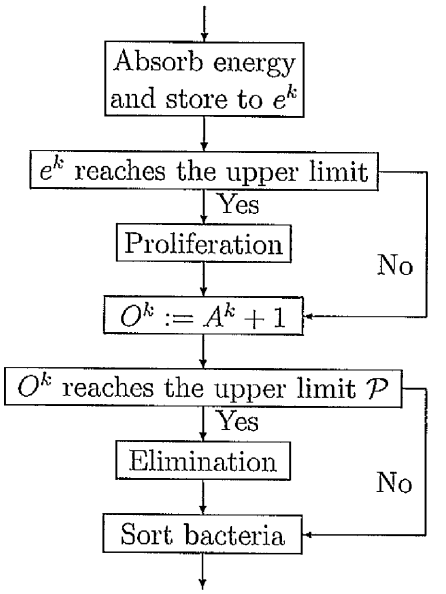


Figure 2.3: The flow chart of reproduction and elimination processes in BFAVP

2.3 Experimental Studies

2.3.1 Parameter setting

In order to evaluate the performance of BFAVP, 13 benchmark functions [9] were adopted in the following experimental studies. In paper [9], 23 benchmark functions were employed, and the number was larger than that offered in many other empirical study papers. It is known that under certain assumptions no single search algorithm is best on average for all problems [92]. If the number of test problems is small, it would be very difficult to make a generalised conclusion. As BFAVP focuses on the optimisation problem in large-scale environment, most benchmark functions in this experiment were selected with high dimensionality. Meanwhile, these 13 benchmark functions covers most difficult cases in the optimisation area.

BFAVP was used for the minimisation of the benchmark functions. Functions $f_1 \sim f_5$ listed in Appendix A.1 are high-dimensional unimodal benchmark functions, used to investigate the convergence rate of each algorithm. Functions

$f_6 \sim f_{10}$ listed in Appendix A.2 are high-dimensional multimodal benchmark functions, which have many local optima. The total number of the local optima of each function increases as its dimension increases. Functions $f_{11} \sim f_{13}$ shown in Appendix A.3 are low-dimensional multimodal benchmark functions [93]. It should be mentioned that the optima of f_6, f_{10}, f_{11}, f_{12}, and f_{13} are not located in the centre of the solution space. The boundaries of the variables, the number of dimensions and the minimum value of each benchmark function are also given in the Appendix A.

In the experimental studies, the performance of BFAVP was compared with that of the following six EAs: GA [25], PSO [35], Fast Evolutionary Programming (FEP) [9], Group Search Optimiser GSO [44], BFA [48], and Adaptive Bacteria Foraging Optimisation Algorithm (ABFOA) [94]. The GA is a optimisation algorithm based on natural selection. Its optimisation process is inspired from biological evolution. PSO is a population based stochastic optimisation technique inspired by social behaviour of bird flocking or fish schooling. GA and PSO are popular algorithms that have been widely applied, and their advantage in computational time has been proven in many studies. FEP uses a Cauchy instead of Gaussian mutation operator in EP as the primary search operator. FEP is faster and reaches smaller fitness values on certain benchmark functions, which has been demonstrated in [9]. GSO is a novel optimisation algorithm proposed in 2009, which is inspired by animal searching behaviour based on the producer-scrounger model. Paper [93] has shown the superiority of GSO over GA, PSO, and FEP under the majority of conditions. Similar as BFAVP, ABFOA is a BIAs which inspired from bacterial foraging behaviour. The step size of the bacteria in ABFOA is regulated by the fitness value. Paper [94] gives detailed comparisons between BFA and ABFOA for several benchmark functions.

Because BFAVP is a varying population-based algorithm, it is fair to compare it with EAs that have been developed using a varying population framework. Such EAs include GAVaPS [80], APGA [81], and PRoFIGA [82]. However, these algorithms were not developed as a generic method: GAVaPS was

designed to solve the benchmark function with less than 3 dimensions, APGA was evaluated on a few benchmark functions that have less than 10 dimensions, and PRoFIGA was designed to meet the demand of Spears' multimodal problems. Moreover, none of the results of GAVaPS, APGA, and PRoFIGA has been reported to be better than that of GA for most of the benchmark functions. Hence, they are not used in the comparative studies. The CMA-ES [85] is typically applied to unconstrained or bounded constraint optimisation problems. The paper aims to reduce the time complexity of the ES on unimodal functions. As a result, the CMA-ES is not compared with the proposed algorithm.

In all experiments, the initial population size of all the algorithms was selected to be 50. For BFAVP, the mean of the bacterial lifespan, μ_{span}, was set to 50, *i.e.*, 50 iterations, and the standard deviation, σ_{span}, was set to 10, *i.e.*, 10 iterations. These two parameters were set up to allow the population size to vary in a more bacterially realistic way, as discussed in Section 2.2.3. However, they were not sensitive to the performance of BFAVP, which will be discussed in Section 2.3.7. The maximum number of run steps taken in a same direction was 4, which will be further discussed in Section 2.3.7 as well. The repelling rate, ζ, is set to 0.2, according to the experiment undertaken, *i.e.*, 20 percent of the bacteria were repelled during the process of quorum sensing. The comparison among different repelling rates will be discussed in Section 2.3.7. The settings for GA were based on the GA Optimisation Toolbox (GAOT) [95], using the normalised geometric ranking as the selection function. The implementation of PSO is described in [35]. For the parameters of PSO, the inertia weight ω was set to 0.73, and the acceleration factors c_1 and c_2 were both set to 2.05, as recommended in [36]. For FEP, the tournament size was set to 10 for selection, as recommended in [9]. In order to make the comparison fair, the tuning of the algorithms did not depend on any assumptions about the objective function. Once the parameters were chosen, they were fixed for all of the experiments. The reason for comparison with GSO is that GSO has been proven better than other EAs for these benchmark functions. The tuning

of GSO parameters followed the recommendation of [44].

Some studies have proposed the use of the t-test to investigate the stochastic properties of an EA. The t-test only assesses whether the means of two groups are statistically different from each other. As in a similar manner to the t-test, as suggested in [9], the mean and standard deviation of the best solution of an algorithm can be used for the purpose of comparison between different algorithms. In order to investigate the stochastic performance and stability of BFAVP, we will present the average mean and standard deviation of the best solution of each algorithm obtained from 50 runs of the algorithm on each benchmark function. In [44], a very large number of runs was set up to investigate the stochastic performance of GSO. To take GSO as an example, it is not necessary to spend time running all algorithms a large number of times for comparison purpose. The performance of BFAVP obtained by selecting a higher number of runs to obtain an average result will be further discussed in Section 2.3.6.

2.3.2 Maximum number of function evaluations

In order to investigate the computational load of each algorithm involved in the comparative studies, the maximum number of function evaluation of each algorithm on each benchmark function was recorded and listed in Table 2.2. The maximal number was suggested in [9], and was obtained once an algorithm fully converged, which means the algorithm was not able to find further improved result after a large number of function evaluations. In the table, the evaluation numbers of GA, PSO, FEP, and GSO are taken from [44]. From the table, it can be seen that the maximum number of function evaluations of BFAVP for each benchmark function is less than or equal to those reported. Concerning the computational load and efficiency of BFAVP, Sections 2.4.1 and 2.4.2 will give further elucidation.

Function	GA*	PSO*	FEP*	GSO*	BFAVP
f_1	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5
f_2	1.5×10^5	1.5×10^5	2.0×10^5	1.5×10^5	1.5×10^5
f_3	1.5×10^5	1.5×10^5	5.0×10^5	1.5×10^5	1.5×10^5
f_4	1.5×10^5	1.5×10^5	2.0×10^6	1.5×10^5	1.5×10^5
f_5	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5
f_6	1.5×10^5	1.5×10^5	9.0×10^5	1.5×10^5	1.5×10^5
f_7	2.5×10^5	2.5×10^5	5.0×10^5	2.5×10^5	2.5×10^5
f_8	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5
f_9	1.5×10^5	1.5×10^5	2.0×10^5	1.5×10^5	1.5×10^5
f_{10}	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5	1.5×10^5
f_{11}	7.5×10^3	7.5×10^3	1.0×10^4	7.5×10^3	7.5×10^3
f_{12}	2.5×10^5	2.5×10^5	4.0×10^5	2.5×10^5	1.5×10^5
f_{13}	1.25×10^3	1.25×10^3	1.0×10^4	1.25×10^3	1.25×10^3

$*$ The numbers are taken from [93] and [9].

Table 2.2: The maximum number of function evaluations of GA, PSO, FEP, GSO, and BFAVP for $f_1 \sim f_{13}$

2.3.3 Experimental studies on unimodal functions

According to the No Free Lunch theorem [92], under certain assumptions there is no single search algorithm that performs best on average for all problems. As evident in Table 2.3, the results of PSO are much better than those of BFAVP on functions f_1 and f_2. This is because the inertial weight of PSO decreases during the optimisation process, which ensures that the search range of the entire population is reduced iteration by iteration. For the type of unimodal functions that have only one optimum, this search strategy helps PSO to find a better solution. However, although this strategy may guarantee convergence when the benchmark function is unimodal, it traps the algorithm in local optima in multimodal functions, which can be seen in the next section.

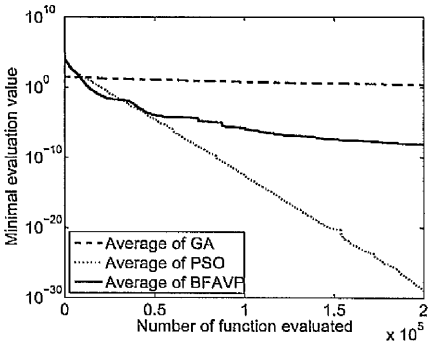
BFAVP outperforms GA on all the unimodal benchmark functions. How-

ever, the random searching characteristic of BFAVP may affect the accuracy of the optimisation result for unimodal functions. Compared with FEP and GSO, BFAVP has nearly the same performance on functions $f_1 \sim f_3$. It is good to see that BFAVP obtains significantly better performance on functions f_4 and f_5. In these cases, the attraction among bacteria offered by quorum sensing in BFAVP is effective, which leads to a fast convergence rate on this type of function that has obvious gradient variation. From the experimental results, it can be concluded that the algorithms that have a step length degression feature, such as PSO, are more suitable for solving unimodal problems with slow gradient variation. The algorithms working with random operators, such as BFAVP, FEP and GSO, are more suitable for solving problems that have obvious gradient variations. The average convergence processes of GA, PSO and BFAVP on $f_1 \sim f_5$ are illustrated in Figures 2.8(a)$\sim$ 2.4(e). When the population size in BFAVP varies, the searching capability is also improved. The improvement of the searching capability can be seen on the later stage of the optimisation in Figures 2.4(d) and 2.4(e).

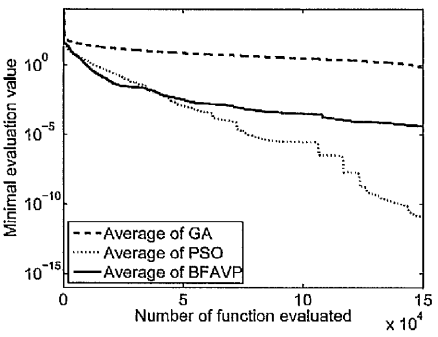
		GA*	PSO*	FEP*	GSO*	BFAVP
f_1	μ	3.1711	$\mathbf{3.6927 \times 10^{-37}}$	5.7×10^{-4}	1.9481×10^{-8}	9.9908×10^{-9}
	σ	1.6621	$\mathbf{2.4598 \times 10^{-36}}$	1.3×10^{-4}	1.9841×10^{-8}	8.7307×10^{-9}
f_2	μ	0.5771	$\mathbf{2.9168 \times 10^{-24}}$	8.1×10^{-3}	3.7039×10^{-5}	5.4701×10^{-5}
	σ	0.1306	$\mathbf{1.1362 \times 10^{-23}}$	7.7×10^{-4}	8.6185×10^{-5}	3.6527×10^{-5}
f_3	μ	7.9610	0.4123	0.3	0.1078	**0.1504**
	σ	1.5063	0.2500	0.5	3.9981×10^{-2}	**0.1221**
f_4	μ	338.5916	37.3582	5.06	49.8359	$\mathbf{7.7345 \times 10^{-4}}$
	σ	361.497.	32.1436	5.87	30.1771	$\mathbf{2.10485 \times 10^{-3}}$
f_5	μ	3.6970	0.1460	**0**	1.6000×10^{-2}	**0**
	σ	1.9517	0.4182	**0**	0.1333	**0**

$*$ The results are taken from [93] and [9].

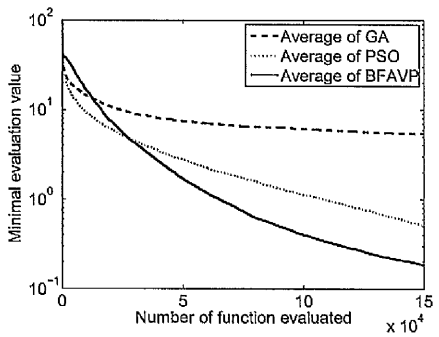
Table 2.3: Average best results of mean (μ) and standard deviation (σ) of GA, PSO, FEP, GSO, and BFAVP on $f_1 \sim f_5$



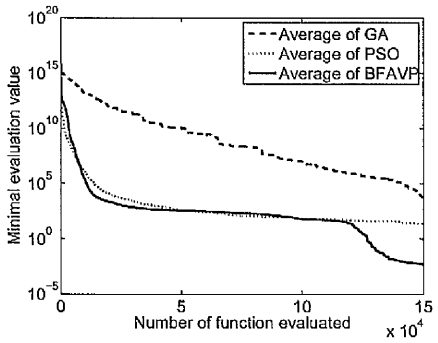
(a) f_1



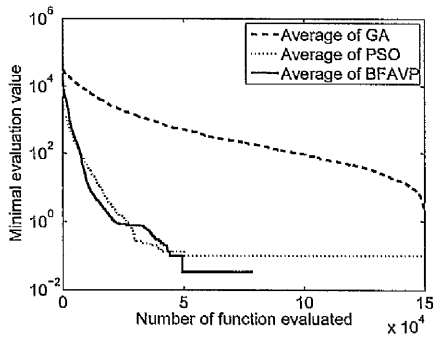
(b) f_2



(c) f_3



(d) f_4



(e) f_5

Figure 2.4: Convergence processes of GA; PSO, and BFAVP on $f_1 \sim f_5$

2.3.4 Experimental studies on multimodal functions

The results listed in Tables 2.4 and 2.5 were obtained from the evaluation of all algorithms on multimodal benchmark functions, which reveal that multimodal problems can be solved effectively by BFAVP because it has both the random search and gradient search capabilities and compromises between them well. In the multimodal functions, the number of local optima exponentially increases in proportion to the increase in the number of dimensions. Therefore, it is difficult for most EAs to solve multimodal functions. These EAs lack on capability of increasing the population diversity because their population size is fixed. As a result, they are very likely to be trapped in local optima. Consequently, the EAs that have a decreasing step length are incapable of finding an accurate solution in the search space of multimodal functions.

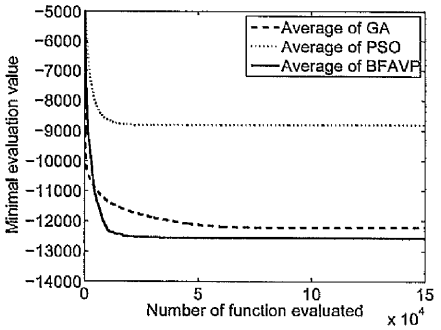
It can be seen from Table 2.4 that BFAVP outperforms most algorithms on benchmark functions $f_6 \sim f_{10}$. BFAVP has nearly the same good performance as GSO on f_6, which has 9.3132×10^{20} local optima. The results show that all the algorithms that have a random-walk mechanism, such as mutation, are suitable for this type of functions. In particular, BFAVP models the quorum sensing behaviour, which allows bacteria to randomly move from local optima to a better position. Accordingly, quorum sensing increases the probability of BFAVP finding the global optimum.

BFAVP also outperforms GA, PSO, and FEP on functions $f_{11} \sim f_{13}$, as shown in Table 2.5. The small standard variance of GSO and BFAVP shows that they always converge to the global optima on these functions. In this case, GSO is slightly more stable than BFAVP on these low-dimensional multimodal problems because the standard deviation of GSO is smaller than that of BFAVP. The average convergence processes of GA, PSO, and BFAVP on $f_6 \sim f_{13}$ are illustrated in Figures 2.5(a)$\sim$ 2.6(c), respectively. As it can be seen from Figure 2.5(b), GA and PSO have potential to converge to better results. However, the experimental results in [9] and [93] indicated that most algorithms, such as EP, ES, FEP, and GSO, had already converged with the suggested number of function evaluations.

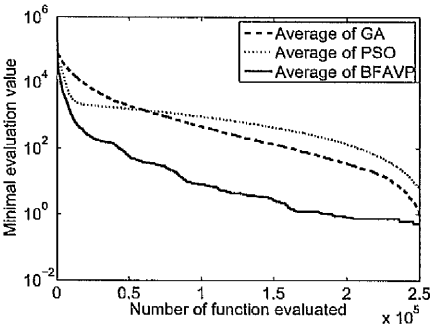
		GA*	PSO*	FEP*	GSO*	BFAVP
f_6	μ	-12566.0977	-9659.6993	-12554.5	-12569.4882	$\mathbf{-12569.4882}$
	σ	2.1088	463.7825	52.6	2.2140×10^{-2}	$\mathbf{1.3082 \times 10^{-2}}$
f_7	μ	0.6509	20.7863	$\mathbf{4.6 \times 10^{-2}}$	1.0179	0.3319
	σ	0.2805	5.9400	$\mathbf{1.2 \times 10^{-2}}$	0.9509	0.3368
f_8	μ	0.8678	1.3404×10^{-3}	1.8×10^{-2}	2.6548×10^{-5}	$\mathbf{1.6325 \times 10^{-5}}$
	σ	0.2805	4.2388×10^{-2}	2.1×10^{-2}	3.0820×10^{-5}	$\mathbf{2.9589 \times 10^{-5}}$
f_9	μ	1.0038	$\mathbf{1.0633 \times 10^{-2}}$	2.6×10^{-2}	3.1283×10^{-2}	2.4649×10^{-2}
	σ	6.7545×10^{-2}	$\mathbf{1.0895 \times 10^{-2}}$	2.2×10^{-2}	2.8757×10^{-2}	4.2953×10^{-2}
f_{10}	μ	4.3572×10^{-2}	3.9503×10^{-2}	9.2×10^{-6}	2.7648×10^{-11}	$\mathbf{1.9596 \times 10^{-9}}$
	σ	5.0579×10^{-2}	9.1424×10^{-2}	6.1395×10^{-5}	9.1674×10^{-11}	$\mathbf{2.0533 \times 10^{-9}}$

$*$ The results are taken from [93] and [9].

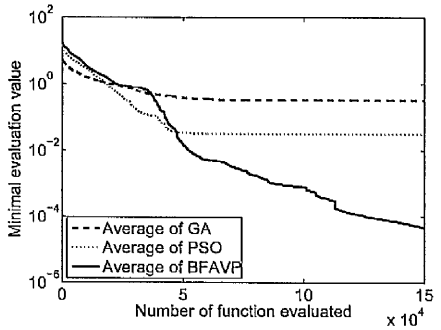
Table 2.4: Average best results of mean (μ) and standard deviation (σ) of GA, PSO, FEP, GSO, and BFAVP on $f_6 \sim f_{10}$



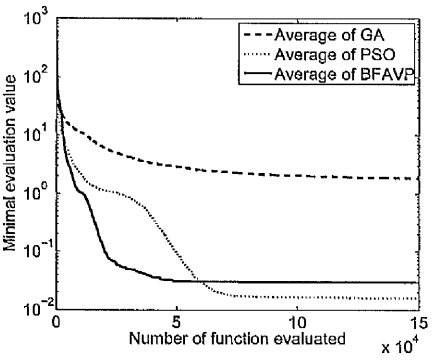
(a) f_6



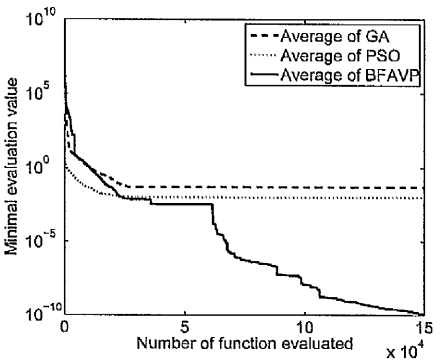
(b) f_7



(c) f_8

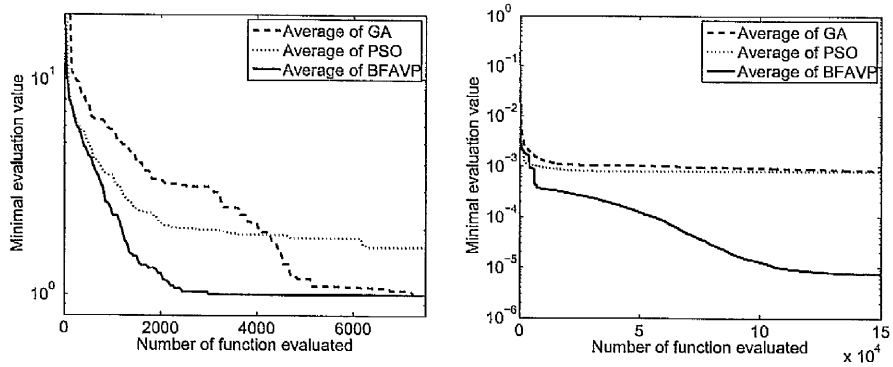


(d) f_9

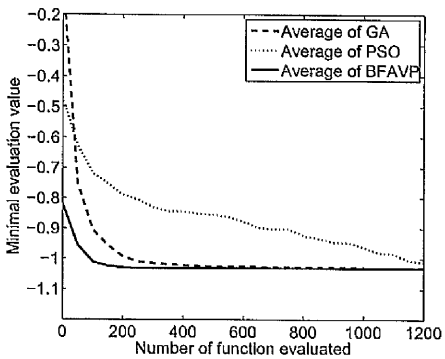


(e) f_{10}

Figure 2.5: Convergence processes of GA, PSO, and BFAVP on $f_6 \sim f_{10}$



(a) f_{11} (b) f_{12}



(c) f_{13}

Figure 2.6: Convergence processes of GA, PSO, and BFAVP on $f_{11} \sim f_{13}$

		GA*	PSO*	FEP*	GSO*	BFAVP
f_{11}	μ	0.9989	1.0239	1.22	**0.9980**	**0.9980**
	σ	4.4333×10^{-3}	0.1450	0.56	**0**	2.5080×10^{-16}
f_{12}	μ	7.0878×10^{-3}	3.8074×10^{-4}	5.0×10^{-4}	4.1687×10^{-4}	$\mathbf{3.7594 \times 10^{-4}}$
	σ	7.8549×10^{-3}	2.5094×10^{-4}	3.2×10^{-4}	$\mathbf{3.1238 \times 10^{-4}}$	4.2768×10^{-4}
f_{13}	μ	-1.0298	-1.0160	-1.03	$\mathbf{-1.0316}$	$\mathbf{-1.0316}$
	σ	3.1314×10^{-3}	1.2786×10^{-2}	4.9×10^{-4}	**0**	5.3284×10^{-16}

∗ The results are taken from [93] and [9].

Table 2.5: Average best results of mean (μ) and standard deviation (σ) of GA, PSO, FEP, GSO, and BFAVP on $f_{11} \sim f_{13}$

2.3.5 The comparison among BIAs

In order to evaluate the improvement of the proposed algorithm, this experiment compared BFAVP with conventional BFA and ABFOA, which is a novel variation of BFA proposed in 2009 [94]. Paper [94] presents the detailed results of ABFOA and the classical BFA using a test-suite of six well-known benchmark functions – Sphere function f_1, Rosenbrock function f_4, Rastrigin function f_7, Griewank function f_9, Ackley function f_8, and Shekel's foxholes function f_{11}. In this section, BFAVP was applied to these benchmark functions with the same number of function evaluations recommended in the paper, which is 5×10^6 evaluations for each benchmark function. The results are listed in Table 2.6.

	No. of FEs		BFOA	ABFOA	BFAVP
f_1	5×10^5	μ	0.084	0.045	$\mathbf{4.6807 \times 10^{-4}}$
		σ	0.0025	0.0061	$\mathbf{1.0281 \times 10^{-4}}$
f_4	5×10^5	μ	58.216	40.212	**15.4971**
		σ	14.3254	**6.5465**	12.1279
f_7	5×10^5	μ	17.0388	**3.3316**	9.7266
		σ	4.8421	**0.5454**	5.8146
f_8	5×10^5	μ	2.3243	0.8038	**0.3966**
		σ	1.8833	0.5512	**0.5103**
f_9	5×10^5	μ	0.3729	0.2414	$\mathbf{3.2041 \times 10^{-3}}$
		σ	0.0346	0.5107	$\mathbf{2.9450 \times 10^{-3}}$
f_{11}	1×10^5	μ	1.0564	0.9998	**0.9980**
		σ	0.0122	0.0017	$\mathbf{7.1036 \times 10^{-10}}$

$*$ The results are taken from [94].

Table 2.6: Average best results of mean (μ) and standard deviation (σ) of BFA, ABFOA, and BFAVP

According to the results, it can be seen that BFAVP performs better than classical BFA on all benchmark functions. The ABFOA improves the chemo-

taxis model of BFA, which leads to the improvement on all benchmark functions as well. However, compared with ABFOA, BFAVP engages more detailed bacterial behaviours into the framework to meet the demands of different optimisation environments. As a result, BFAVP outperforms ABFOA on 5 benchmark functions. In summary, the varying population framework in BFAVP overcomes the limits of the chemotaxis models in BFA and ABFOA.

2.3.6 Robustness and computation time

To further investigate evaluate the robustness of BFAVP, different numbers of runs and function were selected from the three different categories of benchmark functions, *i.e.*, f_1, f_6, and f_{11}. Table 2.7 lists the standard deviation of the best fitness values of BFAVP, obtained using different numbers of runs. The standard deviations exhibit only a small differences between them, which reveals that BFAVP is able to obtain a stable solution after 1.5×10^5 evaluations for f_1 and f_6 and 1.5×10^4 evaluations for f_{11}, which are close to the results shown in Tables 2.3, 2.4, and 2.5, respectively. According to these results, BFAVP is better than the other EAs, at least in the case of 50 runs, and selecting the number of runs to be 50 is reasonable to evaluate the convergence and robustness of BFAVP in comparison with the other EAs.

According to the results presented in Tables 2.3, 2.4, and 2.5, it can be seen that the convergence performances of FEP, GSO and BFAVP, obtained from 50 runs, are similar, and there is a significant difference between those of GA, PSO, and BFAVP. Therefore, a further comparison of BFAVP with GA and PSO is given by considering the computation time taken for high-dimensional benchmark functions. All the experiments were undertaken using a PC with a Core 2 Duo 2.66GHz processor, and computation time was measured in seconds. Tables 2.8 and 2.9 list the computation time consumed by GA, PSO, and BFAVP, respectively, on $f_1 \sim f_{10}$, which was measured when the algorithm reached the pre-set threshold listed in the table. From the results, it can be seen that except for the functions with quadratic surfaces, such as f_1 and f_2, BFAVP is able to perform faster than GA and PSO. The maximal computation

time saving made by BFAVP applied on the benchmark functions can reach up to 87.42% in comparison with GA or PSO. BFAVP is able to make a significant saving in almost all the cases investigated.

	30 runs	50 runs	100 runs	300 runs
f_1	9.4172×10^{-9}	8.7307×10^{-9}	8.0524×10^{-9}	9.1141×10^{-9}
f_6	1.3122×10^{-2}	1.3082×10^{-2}	1.2962×10^{-2}	1.2900×10^{-2}
f_{11}	2.6072×10^{-16}	2.5080×10^{-16}	2.7530×10^{-16}	2.6944×10^{-16}

Table 2.7: Average standard deviation of BFAVP achieved by different number of runs

Time (s)	f_1	f_2	f_3	f_4	f_5
Threshold	0.01	0.01	10.00	100.00	10.00
GA	Fail*	Fail*	25.36	Fail*	18.31
PSO	**0.92**	**3.78**	5.50	36.92	5.27
BFAVP	5.35	5.74	**3.19**	**35.24**	**4.93**
Time reduction	-481.52%	-51.85%	87.42%	4.55%	73.07%

* "Fail" means that the average computation time never reaches the threshold.

Table 2.8: Average computation time (s) consumed to obtain a valid solution with different thresholds on unimodal benchmark functions

To focus on the comparison of computation time between GA, PSO, and BFAVP, different thresholds at 70%, 90%, 99% and 99.99% of the fitness value of function f_6 were set up for the algorithm to reach. Table 2.10 shows the number of function evaluations taken by each algorithm to reach a solution for f_6. Compared with GA and PSO, it can be seen that BFAVP is always the algorithm which reaches the thresholds first. It has also been noted that in some cases, PSO fails to converge to the global optimum of f_6.

Time (s)	f_6	f_7	f_8	f_9	f_{10}
Threshold	-10000	1.00	1.00	1.00	1.0
GA	4.84	21.07	19.67	Fail*	14.54
PSO	Fail*	Fail*	**6.95**	6.37	15.72
BFAVP	**3.96**	**18.35**	7.32	**4.98**	**11.5**
Time reduction	18.18%	12.91%	62.79%	21.82%	26.84%

∗ "Fail" means that the average computation time never reaches the threshold.

Table 2.9: Average computation time (s) consumed to obtain a valid solution with different thresholds on multimodal benchmark functions

	Threshold 1 70%	Threshold 2 90%	Threshold 3 99%	Threshold 4 99.99%
GA	2.1×10^3	5.7×10^3	8.9×10^4	1.3×10^5
PSO	1.3×10^3	Fail*	Fail*	Fail*
BFAVP	$\mathbf{4.1 \times 10^2}$	$\mathbf{3.9 \times 10^3}$	$\mathbf{1.2 \times 10^4}$	$\mathbf{1.0 \times 10^5}$

∗ "Fail" means that the algorithm never reaches the threshold.

Table 2.10: Number of function evaluations required by each algorithm to reach a demand solution on f_6

This experiment gives an insight into the convergence capability of BFAVP by exploring its merits when used with varying population size, which is adaptive to the demand of computational complexity for the reduction of redundant computation. Figure 2.7 illustrates the standard deviation value of BFAVP obtained during the optimisation procedure for function f_6, averaged using 50 runs. The figure shows that during the whole search period, the major search mechanism changed from randomly searching to gradient searching, which can be seen around 1.2×10^5 function evaluations. After that, the optimisation process became stable. The figure shows that it can be seen that the standard deviation not only decreased in the early stage of the optimisation procedure

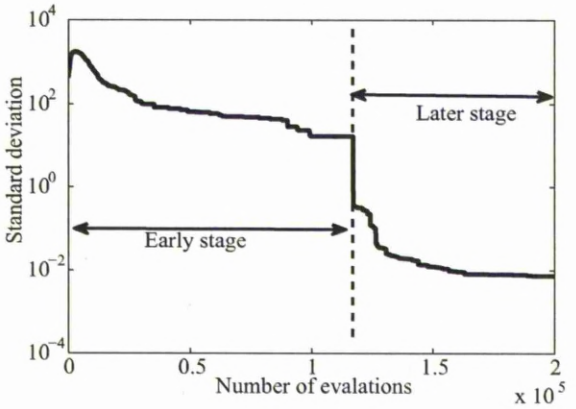


Figure 2.7: The standard deviation of f_6 obtained by BFAVP in 50 runs

but also became nearly invariable at the later stage on the multimodal objective functions. A sudden change of the standard deviation of fitness values during the optimisation process implies that the varying population size is capable of exploring global optima. The rapid change of the standard deviation occurring in the early search stage or immediately after the new optima are explored also imply that BFAVP is able to offer a local search capability. The compromise between the global and local search capabilities is one of the major merits of BFAVP.

2.3.7 Investigation of the parameter settings

According to the previous experiment, it is necessary to understand which parameters of BFAVP would be sensitive to its performance. For this purpose, a set of experiments was undertaken in which three benchmark functions, f_1, f_6, and f_{11}, were selected to evaluate the effect of each parameter. The parameters investigated were the number of runs in a chemotaxis process, percentage of bacteria involved in the attraction and repelling, and the mean and standard deviation of bacterial lifespan, respectively.

In BFAVP, the ratio between tumble and run is an important quantity which could influence the search process. This quantity is controlled by N_c,

which indicates the number of runs in a chemotaxis process. Table 2.11 lists the average best results and standard deviation of functions f_1, f_6, and f_{11} obtained with different N_c. From these results, it can be seen that the variation of N_c has a minor influence on the unimodal benchmark functions. This is due to the fact that, for the unimodal functions, the tumble-run process is the major searching mechanism dominating the whole search process in this case. However, for the multimodal functions, quorum sensing would play as major role in searching. If increasing the number of run steps, the number of quorum sensing would be decreased if the total number of function evaluations is fixed. In the case, quorum sensing would not be able to dominate the search process, but once it applies during the search process, it will cause a significant disturbance to search performance which leads to a large deviation of fitness values and even to an unstable performance of BFAVP.

n_c		f_1	f_6	f_{11}
1	μ	9.0318×10^{-9}	-10778.4387	1.0253
	σ	8.2769×10^{-9}	1978.0344	0.3816
2	μ	8.1712×10^{-9}	-12170.3171	1.0035
	σ	8.7060×10^{-9}	281.9502	3.7655×10^{-3}
3	μ	8.3922×10^{-9}	-12532.8235	**0.9980**
	σ	9.6555×10^{-9}	16.6948	7.1869×10^{-7}
4	μ	9.9908×10^{-9}	**−12569.4882**	**0.9980**
	σ	8.7307×10^{-9}	$\mathbf{1.3082 \times 10^{-2}}$	$\mathbf{2.5080 \times 10^{-16}}$
5	μ	$\mathbf{1.1340 \times 10^{-10}}$	-12569.3084	**0.9980**
	σ	$\mathbf{2.6787 \times 10^{-10}}$	0.1752	5.0517×10^{-16}
6	μ	9.7577×10^{-9}	-12568.0587	**0.9980**
	σ	9.7431×10^{-9}	2.0971	4.4898×10^{-15}

Table 2.11: Average best results of mean (μ) and standard deviation (σ) of functions f_1, f_6, and f_{11} with different number of runs in a chemotaxis process, n_c

Based on the description in Section 2.2, the percentage of bacteria involved in the attraction and repelling of quorum sensing is further discussed here. Different ζ ranging from 0 to 1 were tested on each function. Table 2.12 lists the average best results of mean and standard deviation of functions f_1, f_6, and f_{11} with different ζ. The table shows that with ζ set to be 0.2, BFAVP provides the best results on multimodal functions. For the unimodal functions, the performance is slightly improved by deceasing ζ. However, a smaller ζ also decreases the robustness of the algorithm. As indicated above, in most of the experiments, the ζ has been set to be 0.2.

ζ		f_1	f_6	f_{11}
0	μ	$\mathbf{1.0211 \times 10^{-10}}$	-8904.9134	0.9980
	σ	7.4112×10^{-9}	1008.9575	3.1419×10^{-13}
0.2	μ	9.9908×10^{-9}	**−12569.4882**	**0.9980**
	σ	8.7307×10^{-9}	$\mathbf{1.3082 \times 10^{-2}}$	$\mathbf{2.5080 \times 10^{-16}}$
0.4	μ	9.9519×10^{-9}	-12539.6324	0.9980
	σ	3.1270×10^{-9}	92.1576	6.4218×10^{-14}
0.6	μ	8.0917×10^{-9}	-12432.0975	0.9980
	σ	$\mathbf{1.7307 \times 10^{-9}}$	253.9706	9.7922×10^{-12}
0.8	μ	7.7725×10^{-9}	-11052.2785	0.9980
	σ	8.7307×10^{-8}	531.4854	5.6557×10^{-11}
1	μ	3.9058×10^{-9}	-9247.5469	0.9980
	σ	3.7307×10^{-8}	691.8003	7.8491×10^{-10}

Table 2.12: Average best results of mean (μ) and standard deviation (σ) of functions f_1, f_6, and f_{11} with different percentage of bacteria involved in the attraction, ζ

In BFAVP, two parameters, mean and standard deviation of bacterial lifespan, are related to the elimination behaviour. Tables 2.13 and 2.14 list the average best results of the mean (μ) and standard deviation (σ) of functions f_1, f_6, and f_{11}, respectively, with a different mean (μ_{span}) and standard deviation (σ_{span}) in equation (2.2.9). In Table 2.13, the standard deviation of the

bacterial lifespan is set to be 10. In Table 2.14, the mean bacterial lifespan is set to be 50. From these two figures, it is understood that these two parameters are not sensitive to the performance of BFAVP.

μ_{span}		f_1	f_6	f_{11}
30	μ	9.9514×10^{-9}	$\mathbf{-12569.4882}$	$\mathbf{0.9980}$
	σ	8.5284×10^{-9}	1.8147×10^{-2}	3.0975×10^{-16}
50	μ	9.9908×10^{-9}	$\mathbf{-12569.4882}$	$\mathbf{0.9980}$
	σ	$\mathbf{8.7307 \times 10^{-9}}$	1.3082×10^{-2}	2.5080×10^{-16}
70	μ	$\mathbf{9.8759 \times 10^{-9}}$	$\mathbf{-12569.4882}$	$\mathbf{0.9980}$
	σ	8.9174×10^{-9}	$\mathbf{1.1270 \times 10^{-2}}$	$\mathbf{2.2785 \times 10^{-16}}$

Table 2.13: Average best results (μ) and standard deviation (σ) of functions f_1, f_6, and f_{11} with different bacterial lifespan, μ_{span}

σ_{span}		f_1	f_6	f_{11}
5	μ	$\mathbf{9.9102 \times 10^{-9}}$	$\mathbf{-12569.4882}$	$\mathbf{0.9980}$
	σ	$\mathbf{8.5913 \times 10^{-9}}$	1.9134×10^{-2}	2.5469×10^{-16}
10	μ	9.9908×10^{-9}	$\mathbf{-12569.4882}$	$\mathbf{0.9980}$
	σ	8.7307×10^{-9}	$\mathbf{1.3082 \times 10^{-2}}$	$\mathbf{2.5080 \times 10^{-16}}$
15	μ	9.9780×10^{-9}	$\mathbf{-12569.4882}$	$\mathbf{0.9980}$
	σ	9.1447×10^{-9}	1.6324×10^{-2}	2.9575×10^{-16}

Table 2.14: Average best results (μ) and standard deviation (σ) of functions f_1, f_6, and f_{11} with different standard deviation of lifespan, σ_{span}

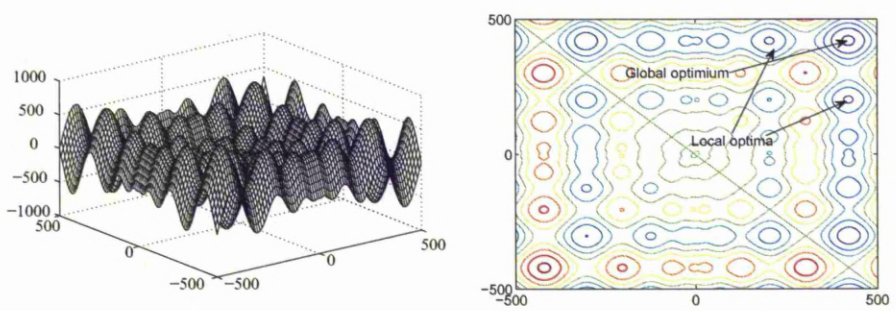
2.4 Bacterial Colonial Behaviours in Optimisation

In biology, the colonial behaviour refers to several individual organisms of the same species living closely together. A bacterial colony is a cluster of

organisms reproduced usually from a single bacterium. Based on the simulation studies of colon behaviours in BFAVP, the advantages of BFAVP can be analysed considering three aspects, *i.e.*, variation of population size, efficiency of energy absorption, bacterial migration, and bacterial corpse distribution. In order to demonstrate the bacterial colony behaviours visually, BFAVP was applied on f_6 in a 2-dimensional case as follows:

$$f_6(x) = -\sum_{i=1}^{2} \left(x_i \sin\left(\sqrt{|x_i|}\right)\right). \tag{2.4.1}$$

where f_6 is a multimodal function as illustrated in Figure 2.8, which has a nourish distribution akin to that of the practical environment.



(a) f_1 (b) f_2

Figure 2.8: The landscape (a) and contour line (b) of f_6

2.4.1 Variation of population size

As described in the previous section, the population size varies in BFAVP. Bacteria absorb energy from the nutrient environment and proliferate if enough energy is gained. A bacterium is eliminated once it reaches the limitation of its lifespan.

When the bacteria are moving in the search space, a measure, optimisation

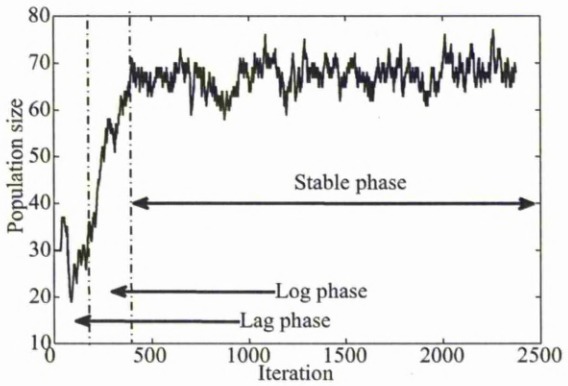


Figure 2.9: The varying population size in BFAVP

efficiency ξ, can be determined as follows:

$$\xi = \frac{\hat{S}^k}{S^k}, \tag{2.4.2}$$

where S^k denotes the population size in the k^{th} iteration, and $\hat{S}^k$ is the number of the bacteria that have a better fitness value in k^{th} iteration than in the $(k-1)^{\text{th}}$ iteration. $\hat{S}^k$ can be expressed as:

$$\begin{aligned} \hat{S}^k &= \sum_{p=1}^{S^k} Q_p^k, \\ Q_p^k &= \begin{cases} 1, & F(X_p^k) < F(X_p^{k-1}), \\ 0, & \text{otherwise}, \end{cases} \end{aligned} \tag{2.4.3}$$

where $F(X_p^k)$ indicates the fitness value of the p^{th} bacterium at the k^{th} iteration, and S^k indicates the population size of k^{th} iteration. As the population changes each iteration, the optimisation efficiency ξ also changes.

The objective function described by (2.4.1) is a 2-dimensional multimodal function. Therefore, the computation is not as complex as that undertaken in the previous simulation studies. Thus, the initial population size was reduced from 50 to 20. The limit of population size was set to 150. Figure 2.9 shows the variation of population size within 2500 iterations. As mentioned in Section

1.3.3, *E. coli* growth follows three stages. In the lag phase, bacteria do not immediately reproduce, and the population size remains constant. The population then enters the logarithmic phase, in which cell numbers increase at a logarithmic rate. The logarithmic phase of bacterial growth is followed by the stationary phase, in which the population size remains constant, even though some cells continue to divide and others begin to die. From this figure, the lag phase, logarithmic phase and stationary phase can be explicitly observed. Figure 2.10 illustrates the optimisation efficacy, ξ, which follows (2.4.2). According to this figure, the population size is stabilised at the later stage because of that BFAVP is self-adaptive according to the complexity of this specified objective function. As a result, it is unnecessary to set the population to a constant value. Otherwise, the optimisation efficiency will be restricted as discussed in the next section.

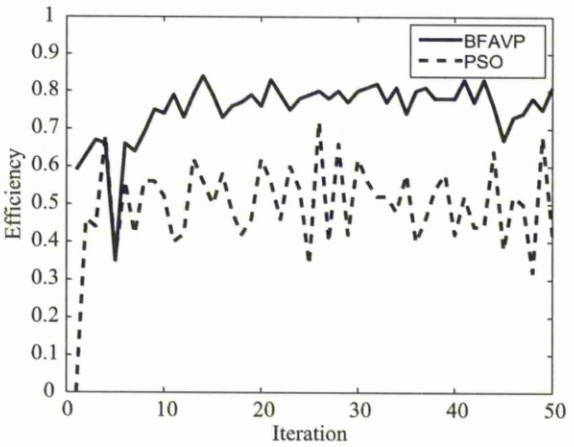


Figure 2.10: Optimisation efficiency

2.4.2 Efficiency of energy absorption

As described in [48], bacteria search for energy in a way that maximises their efficiency. Suppose E is the quantity of energy absorbed by an individual

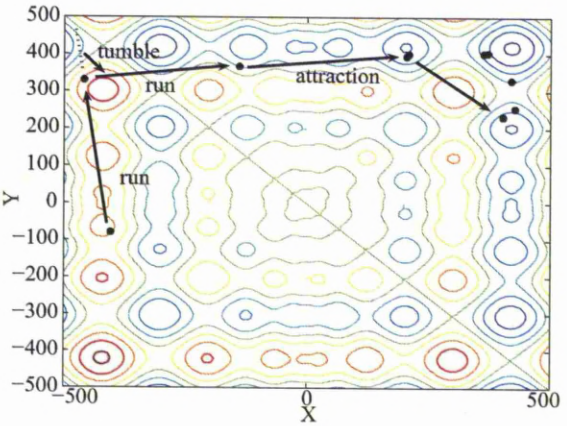


Figure 2.11: A bacterial locus on contour line

during a time interval, T. The aim of bacterial foraging is to maximise the function of:

$$\eta = \frac{E}{T}, \tag{2.4.4}$$

where η is the foraging efficiency. Thus, an algorithm inspired by bacterial chemotaxis behaviour should be highly efficient in searching. This feature can be seen from Figure 2.11, which illustrates the locus of a single bacterium on the function f_6. Although the rotation angle is randomly generated, the direction of the run step is most likely in the gradient-decreasing direction due to the equation (2.2.4). Even if the bacterium is trapped into local optima, the repelling process of quorum sensing can still provide opportunities for the bacterium to escape.

To measure the foraging efficiency η, GA, PSO, and BFAVP were applied on function f_6, respectively. The energy of the p^{th} bacterium at the k^{th} iteration is proportional to the fitness value, $F(X_p^k)$. The sum of the energy in the whole tumble-run process, E^p, can be expressed as,

$$E_p = \sum_{k=1}^{\mu_{\text{span}}} F(X_p^k), \tag{2.4.5}$$

where μ_{span} is the mean of bacterial lifespan. Therefore, equation (2.4.4) can be rewritten as:

$$\eta = \frac{\sum_{p=1}^{\bar{S}} E_p}{\bar{S}\mu_{\text{span}}} = \frac{\sum_{p=1}^{\bar{S}} \sum_{k=1}^{\mu_{\text{span}}} F(X_p^k)}{\bar{S}\mu_{\text{span}}}. \tag{2.4.6}$$

In BFAVP, $\bar{S}$ is the mean value of the population size for the whole optimisation process.

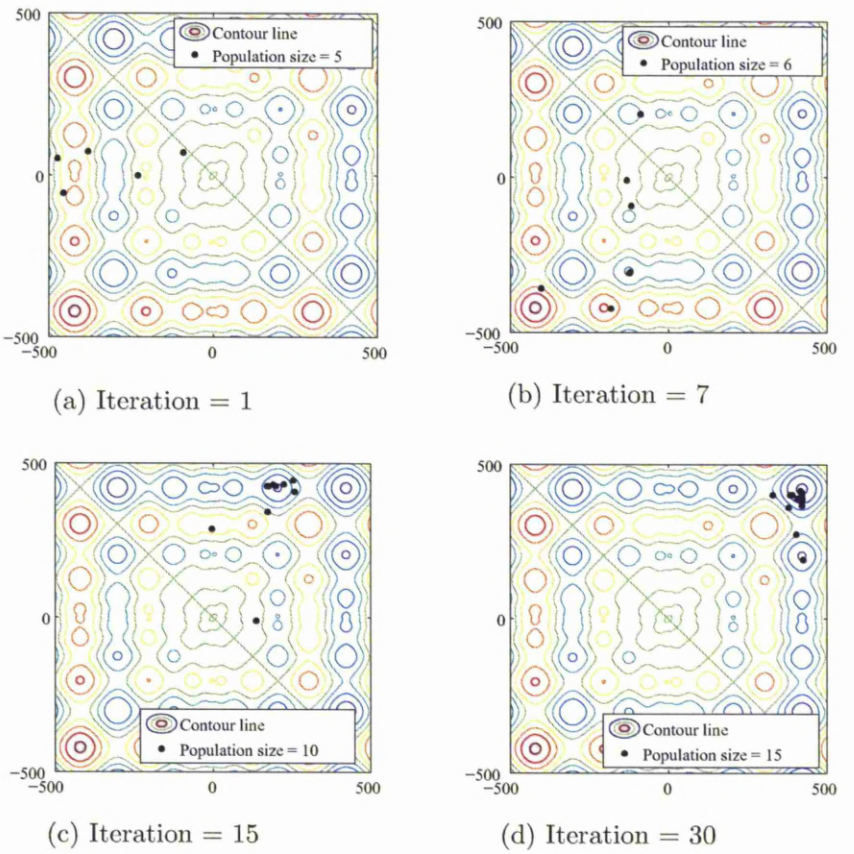
The average foraging efficiencies of GA, PSO, and BFAVP, obtained in 50 runs, were 657.8561, 662.6191 and 794.3517, respectively. It can be seen that the foraging efficiency of BFAVP is higher than that of GA and PSO, *i.e.*, BFAVP is more suitable for multimodal problems.

2.4.3 Bacterial migration

Bacterial migration denotes the movement of bacteria from one location to another, sometimes over a long distance or in large groups. The mechanism of chemotaxis senses the distribution of nutrients in the environment and determines in which direction the concentration increases in the fastest way. However, as the size of a bacterium body is small, it is hard for bacteria to detect the gradient of nutrients effectively. Therefore, bacteria use proliferation and elimination to accelerate migration speed. By introducing the framework of varying population, in particular the proliferation and elimination behaviour, BFAVP has less chance of being trapped into local optima.

In order to analyse the behaviour of bacterial migration, an experiment was undertaken to monitor the position of each bacterium during the optimisation process. In this case, for bacteria to be easily and clearly observed, the initial population size of BFAVP was set to 5, as a small number. Figure 2.12 illustrates four stages in the optimisation process. In the initial iteration, five bacteria were randomly placed in the nutrient environment, which were far away from the global optimum located at the position of (420.9687.420.9687). In the 7^{th} iteration, the population size was increased by 1. The bacteria were still far away from the global optimum and some of them were even trapped in

local optima. However, in the 15^{th} iteration, most of the bacteria converged to a local optimum near the global optimum. Nutrient distributed at this location was richer than elsewhere apart from the global optimum. As a result, the population size began to increase. Finally, in the 30^{th} iteration, the population size increased to 20. Most of the bacteria converged to the global optimum and began rapid proliferation.



(a) Iteration = 1

(b) Iteration = 7

(c) Iteration = 15

(d) Iteration = 30

Figure 2.12: The bacterial migration during the foraging process

According to Figure 2.12, it can be seen that BFAVP has an ability to adjust the population size according to the distribution of nutrients. This ability is represented by bacterial migration. When most of the bacteria are trapped

in local optima where the density of nutrients is not high, the population size rises slowly. When the bacteria converge to a small region around the global optimum, the population size will increase greatly due to their proliferation, and each bacterium will then start a subtle search for the global optimum.

2.4.4 Bacterial corpses and local optima

Bacteria search for food according to the information of the nutrient distribution throughout the whole foraging process. This information updates iteration by iteration. In an environment where there are multiple local optima, a bacterium tends to stay in one of the optima if an alterative route is not discovered. Since the elimination mechanism is introduced in BFAVP, those bacteria which did not gain enough energy are eliminated during the optimisation process. With the quorum sensing mechanism which attracts or repels bacteria, the "corpses" of these bacteria are more likely to be found around the local optima or global optimum. As a result, BFAVP is able to detect the local optima in a single run. Thanks to this advantage, the BFAVP is suitable to be applied on dynamic environment. Local optima have the potential to replace the global optima after the environment change. Figure 2.13 illustrates the bacterial "corpses" and local optima along the contour line of function f_6. From this figure, several local optima are detected by the "corpses" positions.

2.5 Conclusion

This chapter has presented a novel optimisation algorithm, BFAVP, which is inspired by five underlying mechanisms of bacterial foraging behaviours: chemotaxis, metabolism, proliferation, elimination and quorum sensing. It is more biologically realistic than BFA. In order to establish a framework of varying population, indexes of bacterial energy and bacterial age have been introduced to BFAVP to gauge the searching capability and life cycle of an individual, thus simulating bacterial metabolism, proliferation and elimination behaviours. The algorithm has also incorporated chemotaxis behaviour with

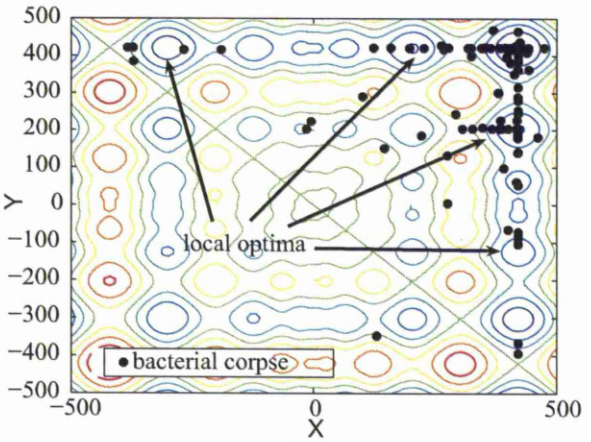


Figure 2.13: Bacterial corpses and local optima on contour line

the quorum sensing phenomenon. The former enables the gradient searching capability, which ensures that the bacterium always moves to a better position than the previous step. The latter plays a role in controlling the diversity of the bacterial population. Attraction and repelling of quorum sensing behaviour have been adopted in BFAVP. When quorum sensing happens inside a single bacterial species, it accelerates the convergence speed. When quorum sensing happens among disparate species, it prevents bacteria from being trapped into local optima. This feature of quorum sensing has significantly enhanced the global search ability of BFAVP.

Experimental studies have shown that thanks to this framework, BFAVP makes the population size adaptive to different types of benchmark functions. BFAVP also overcomes the lack of population diversity, which most EAs suffer from. It has also offered better computation efficiency than other EAs. With flexible operation in quorum sensing, BFAVP is more suitable for high-dimensional multimodal functions than other EAs. Moreover, BFAVP performs stably on unimodal functions, and always obtains better results than GA, PSO, FEP, GSO, BFA, and ABFOA on most of the multimodal benchmark functions, especially with high-dimensional functions, with much less computation time

required.

The research has comprehensively presented the analysis of BFAVP performance via experimental studies on 13 benchmark functions in comparison with GA, PSO, FEP, GSO, FEP, and ABFOA and also by discussions of its behaviours from the bacterial point of view. As these benchmark functions cover almost all the cases of complex optimisation problems, it can be concluded that BFAVP has great potential to be applied to real-world problems, which will be demonstrated in Chapter 4.

Chapter 3

Paired-bacteria Optimiser

Evolutionary Algorithms (EAs) are notorious for their intensive computation as a result of the large amount of evaluations of the objective function required by all individuals. Instead of adopting the commonly used population-based computation, a Paired-bacteria Optimiser (PBO) is proposed in this chapter that is based on a single bacterium foraging model. Different from most EAs, PBO has only one pair of individuals in a population: the primary individual and the pseudo individual. This algorithm is based on the computation of the pseudo gradient from these two individuals. To evaluate the performance, PBO has been tested on several sets of benchmark functions. The experimental results show that it has a better performance than Genetic Algorithm (GA) and Particle Swarm Optimiser (PSO), especially in a dynamic environments.

3.1 Introduction

In the past few decades, EAs have been broadly investigated, and a large number of variants have been developed to solve real-world engineering problems. However, some of the problems can not be solved using large population size-based algorithms because of their computational complexity [11]. Due to the uncertainty in association with the random search that is undertaken in the optimisation process, most EAs introduce unnecessary computation in cal-

culating poorly performing individuals, especially in high-dimensional cases. Work presented in [10] shows that the higher the dimension of the objective function is, the more uncertainties an EA has to face. As a result, only few EAs, such as Estimations of Distributions Algorithms (EDA), Extended Compact GA (ECGA), and Chromosome Compression GA (CCGA), are able to handle complex combinatorial problems that involve a large number of variables [96]. In these algorithms, EDA generates candidate solutions by combining and modifying existing solutions in a stochastic way; ECGA considers that the problem variables can be arranged in clusters of related variables; and CCGA employs chromosome compression and local search strategies to reduce the the dimension of search space.

By studying the relationship between the population size and computational complexity, some EAs have been proposed to overcome the drawback mentioned above. A Particle-Pair Optimiser (PPO) was introduced in requantisation codebook design for grey scale image re-compression [97]. PPO is a novel algorithm proposed for vector quantisation in image coding based on conventional PSO. Experimental results have demonstrated that PPO performs better with shorter computational time and smaller compression distortion in comparison to other conventional vector quantisation algorithms. The experimental results have also shown that PPO can produce a higher peak signal-to-noise ratio (PSNR) value [98] with less transmission delay than other EAs.

However, the major search method of PPO is improved from that of PSO, which means that the algorithm is also limited by the weak side of PSO. In the past decade, some variants of EAs are proposed based on the gradient search method [99][100][101]. Gradient-based search methods are known to be very efficient, especially for cases in which the surface of the solution space is relatively smooth with few local minima [102]. In cases where the solution space lacks this relative uniformity, gradient-based methods may be easily trapped in local minima that are commonly found on rough or complex surfaces. Like the gradient-based approach, most EAs have their own significant weakness.

This stems from the fact that despite reliability, solutions are often not optimal. Furthermore, both methods are known to converge very slowly. Thus, it is necessary to combine random searching into pure gradient searching methods [103][104]. Recently, a Pseudo-gradient-based Evolutionary Programming (PGEP) was proposed based on the study of gradient searching in optimisation [105]. In this algorithm, the pseudo-gradient is introduced to indicate the maximal increase direction of the objective function, which accelerates the convergence towards the best gradient direction. The pseudo-gradient is calculated on the basis of the finite differences of the variation of candidate solution and the corresponding finite differences of their fitness function [106]. In the experimental studies, PGEP shows its merits in relation to gradient-based benchmark functions.

Based on previous research, a PBO is proposed in this chapter, which has only a pair of individuals in a population. The two individuals undertake different tasks. The primary individual plays a role in datum mark, which provides a reference for the next movement of the individuals. The pseudo individual proceeds on a random walk around the primary individual, which provides the gradient information around the primary individual. Occasionally, the pseudo individual is randomly placed in a distant location to increase the diversity of the optimisation. To accelerate the convergence speed of PBO, a simplified quorum sensing method is introduced, which is improved from that of the attraction behaviour in PSO. The simplified quorum sensing can either attract the bacterium to the best fitness location or exchange the dimensional information it has experienced.

To evaluate PBO, the experimental studies employs four sets of benchmark functions, which include high-dimensional unimodal functions, high-dimensional multimodal functions, low-dimensional multimodal functions, and dynamic functions. The experimental results show that the proposed PBO has a more effective searching ability compared to GA and PSO, with a fewer number of evaluations. The results on dynamic benchmark functions illustrate that PBO can rapidly adapt to the environment change rapidly, due to its micro

population size.

The rest of the chapter is organised as follows. Section 3.2 introduces the inspiration from bacterial foraging behaviours and the mathematical models of PBO. The experimental studies of the proposed PBO are presented in Section 3.3 with descriptions of the benchmark functions. The research work of PBO is concluded in Section 3.4.

3.2 The PBO Algorithm

In PBO, there are two individuals in each iteration. The primary individual performs the major search task. A pseudo individual is randomly placed close to primary individual. Pseudo-gradient is calculated on the basis of the distance and the corresponding evaluation value between primary individual and pseudo individual. The position of primary individual is updated based on the value of pseudo-gradient. Meanwhile, due to the lack of population size, a simplified quorum sensing is introduced to accelerate the convergence speed. The mathematical models involved in PBO are pseudo-gradient searching and simplified quorum searching.

3.2.1 Pseudo-gradient searching

In the PGEP [105], if the objective function F is differentiable, for an n-dimensional parameter optimisation problem, a conventional gradient, $\overrightarrow{g}$, of its objective function is defined as:

$$\overrightarrow{g}(\overrightarrow{X}) = \nabla F(\overrightarrow{X}) = \left(\frac{\partial F}{\partial x_1}, ..., \frac{\partial F}{\partial x_n}\right), \tag{3.2.1}$$

which indicates the steepest increase direction of the objective function at a point, $\overrightarrow{X}$ in the search space, which is an essential property of gradient methods. By studying the mechanism of conventional gradient, a pseudo-gradient $\overrightarrow{g}_{\mathrm{p}}$, is defined as follows.

Assume X_p, $X_p \in \mathbb{R}^n$, $X_p = (x_{p,1}, x_{p,1}, ..., x_{p,n})$, is a point in the search space of a minimisation problem, and it moves to a new point, X_q; if $F(X_p) < F(X_q)$,

then PGEP defined that changing direction from X_p to X_q, as positive, and the pseudo-gradient $\overrightarrow{g}_{\mathrm{p}}(X_p, X_q)$ is denoted as:

$$\overrightarrow{g}_{\mathrm{p}}(X_p, X_q) = (\mathrm{dir}(x_{p,1}, x_{q,1}), \mathrm{dir}(x_{p,2}, x_{q,2}), ..., \mathrm{dir}(x_{p,n}, x_{q,n})), \tag{3.2.2}$$

where

$$\mathrm{dir}(\mathrm{x_{p,1}}, \mathrm{x_{q,1}}) = \begin{cases} 1 & x_{q,i} > x_{p,i}, \\ 0, & -x_{q,i} > x_{p,i}, \quad (i = 1, ..., n); \\ -1, & x_{q,i} < x_{p,i}. \end{cases} \tag{3.2.3}$$

if $F(X_q) < F(X_p)$, then PGEP defined that changing direction as negative and the pseudo-gradient $\overrightarrow{g}_{\mathrm{p}}(X_p, X_q)$ is denoted as:

$$\overrightarrow{g}_{\mathrm{p}}(X_p, X_q) = (0, ..., 0), \tag{3.2.4}$$

Similar to $\overrightarrow{g}$, $\overrightarrow{g}_{\mathrm{p}}$ is able to indicate a good search direction based on the latest two points in the search space. From the equations above, it can be found that if the i^{th} element in $\overrightarrow{g}_{\mathrm{p}}(X_p, X_q)$ is not zero, a better solution of the objective function would be found at the next step following the direction on the i^{th} dimension. Otherwise, the search direction at that point should be changed to a randomly selected direction.

In traditional gradient optimisation algorithms, such as the Newton method [107], the convergence depends on the value of gradient. Therefore, in PBO, the velocity of the bacterium is determined by pseudo gradient as well. The position of a bacterium at the k^{th} iteration is defined as $X^k = (x_1^k, x_2^k, ..., x_n^k) \in \mathbb{R}^n$, where x_i^k is the value of the X^k on the i^{th} dimension, and n is the number of dimensions. Meanwhile, a pseudo individual $\widetilde{X}$ accompanies to X at each iteration as a mutation of individual X. A random dimension, $l, l \in \{1, 2, ..., n\}$, is chosen for the mutation dimension. The position of the pseudo individual $\widetilde{X}$ at the k^{th} iteration is expressed as:

$$\widetilde{X}^k = X^k + \delta D^k, \tag{3.2.5}$$

where $\delta D^k = (0, ..., 0, d_l^k, 0, ..., 0)$, and d_l^k indicates a value randomly chosen between the lower boundary and the upper boundary of the searching space in

the l^{th} dimension, which is denoted as:

$$d_l^k = c_c r_1 (B_{\text{up}_l} - B_{\text{lo}_l}), \tag{3.2.6}$$

where r_1 is an uniformly distributed number in $[-1, 1]$, and B_{lo_l} and B_{up_l} denotes the upper and lower boundary of the l^{th} dimension, respectively. A two-value coefficient c_c is introduced to extend the searching range, which plays a role equivalent to maintaining the diversity of the population, as adopted in conventional EAs. When c_c equals a small value, the d_l^k is generated as a small value. As a result, if the distance between the primary individual and the pseudo individual is also small enough, then the distance is the partial gradient in the l^{th} dimension. When c_c equals a large value, the d_l^k is generated as a large value as well. Then the pseudo individual performs a random placement in the searching space. If the pseudo individual is placed out of the feasible space on the l^{th} dimension, it will be placed on the boundary on that dimension.

Set $g_l^k(\widetilde{X}^k, X^k)$ as an alternative format of pseudo gradient along the l^{th} dimension at the k^{th} iteration, as follows.

$$g_l^k(\widetilde{X}^k, X^k) = \frac{F(\widetilde{X}^k) - F(X^k)}{\widetilde{x}_l^k - x_l^k}, \tag{3.2.7}$$

where $F(X^k)$ and $F(\widetilde{X}^k)$ are the evaluation values of X^k and $\widetilde{X}^k$, respectively.

At each iteration, the velocity of the chemotaxis is expressed as:

$$Vel_{\text{c}}^k = (0, ..., g_l^k(\widetilde{X}^k, X^k), ..., 0). \tag{3.2.8}$$

3.2.2 Simplified quorum sensing

The velocity of the primary individual is donated by Vel^k, $Vel^k \in \mathbb{R}^n$, $Vel^k = (vel_1^k, vel_2^k, ..., vel_n^k)$, where vel_i^k is a component of the V^k on the i^{th} dimension at the k^{th} iteration. The velocity is updated at the k^{th} iteration as follows,

$$Vel_{\text{q}}^k = \gamma(Vel^k + r_2(P_{\text{g}}^k - X^k)), \tag{3.2.9}$$

where γ indicates the inertia weight, which is fixed in each iteration, P_{g}^k indicates the position of the best individual form the past k iterations, and r_2 is

a random number, $r_2 \in [0, 1]$. The l^{th} elements vel_l^k in Vel^k is in the range of $[0, vel_l^{\max}]$, where $vel_l^{\max}$ is the maximal velocity of individual X along the l^{th} dimension, which is scaled proportionally to the range of the search boundary of that dimension.

Although PBO has a small computation complexity, its convergence speed in the early stage is still slow, which is caused by the lack of information about the benchmark function in this stage. By studying the relationship between the gradient in each dimension and the optimum, it was noted that the individual located near the optimum has a smaller gradient [108]. As the PBO records the last minimal gradient, the direction of the convergence can be predicted by this information. The learning speed of the bacterium is updated as:

$$Vel_{\text{s}}^k = \begin{cases} r_3(X^k \ominus \widetilde{x}_l^k), & F(\widetilde{X}_l^k) < F(X^k), \\ 0, & \text{otherwise}, \end{cases} \tag{3.2.10}$$

where $r_3 \in [0, 1]$ is an uniformly distributed random number, and $\mathbf{a} \ominus b$ means a number b ($b \in \mathbb{R}$) is subtracted from each element in $\mathbf{a}$. Hence, the position of the primary individual is updated as:

$$X^{k+1} = X^k + Vel_{\text{c}}^k + Vel_{\text{q}}^k + Vel_{\text{s}}^k. \tag{3.2.11}$$

This process is continually performed until a termination criterion is reached. Table 3.1 lists the pseudo code of PBO, and the overall operation of PBO is illustrated in Figure 3.1. The notations used in PBO is listed in Appendix B.2.

Initialise the position of the primary individual and pseudo individual;
Initialise the velocity of the primary individual;
Evaluate the fitness of the primary individual;
Set g_p^0 as the guessed value of global optimum;
Set $k := 1$;
WHILE(the termination conditions are not met)
Select a dimension l randomly;

Pseudo individual:	Place the pseudo individual to position $\widetilde{X}^k$ by equation (3.2.7); Evaluate the fitness value of $\widetilde{X}^k$;
Velocity updating:	Calculate the pseudo gradient along the l^{th} dimension using equation (3.2.6); Calculate the velocity of primary individual by equation (3.2.9);
Gradient updating:	Exchange the gradient information on each dimension by equation (3.2.10);
Generation:	Update the position of the primary individual by equation (3.2.11); Update g_p^k if the fitness of current individual is better;

$k := k + 1$
END WHILE

Table 3.1: Pseudo code of PBO

According to the pseudo code, the major difference between PBO and other EAs is the population size framework. Most EAs are based on a large population size, which requires a series of function evaluations in each iteration. However, there are only two function evaluations in PBO for primary individual and pseudo individual, respectively. Meanwhile, different from swarm intelligence algorithms, the convergence of PBO depends not only on the communication between individuals, but also on the searching mechanism of each individual. Thanks to the pseudo-gradient, the primary individual is able to perform the search independently without sharing the information with the pseudo individual.

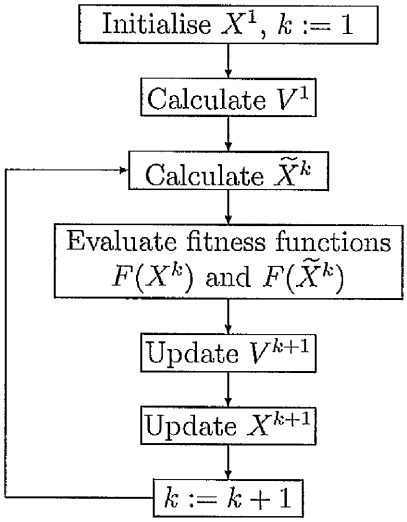


Figure 3.1: The flow chart of PBO

3.3 Experimental Studies

3.3.1 Benchmark functions

In order to evaluate the performance of PBO, four sets of benchmark functions were selected from the research of [9]. These benchmark functions are listed in Appendix A. Function $f_1 \sim f_{10}$ are same as the functions evaluated in Section 2.3. Moreover, three functions ($f_{11} \sim f_{13}$) with error interference are adopted in this experiment. Among the functions, $f_6 \sim f_{10}$ are unimodal functions where the number of local optima increases exponentially with the number of problem dimensions. These functions have been widely tested in [109] and [110]. The number of dimensions, the feasible solution space, and the $f_{\min}$ of these functions are also listed in Appendix A.

3.3.2 Parameter setting

In the experimental studies, PBO was evaluated on the above benchmark functions in comparison with GA [25], PSO [35], Fast Evolutionary Program-

ming (FEP) [9], Group Search Optimiser (GSO) [44], and Bacterial Foraging Algorithm with Varying Population (BFAVP). The number of evaluations for PBO in a compete optimisation process is listed in Table 3.2, which was suggested in [9]. The evaluation numbers of other algorithms were taken from [44], and listed in Table 2.2. PBO was designed to meet the demands of a fast changing environment; thus, the maximal numbers of function evaluations for PBO in each benchmark function are less than those of other algorithms.

Function	Number of evaluations	Function	Number of evaluations
f_1	5.0×10^4	f_9	5.0×10^4
f_2	5.0×10^4	f_{10}	5.0×10^4
f_3	5.0×10^4	f_{11}	2.5×10^3
f_4	5.0×10^4	f_{12}	1.0×10^5
f_5	5.0×10^4	f_{13}	1.0×10^3
f_6	5.0×10^4	f_{14}	1.0×10^5
f_7	5.0×10^4	f_{15}	1.0×10^4
f_8	5.0×10^4	f_{16}	1.0×10^3

Table 3.2: The maximal number of function evaluations of PBO for $f_1 \sim f_{16}$

The parameters of PBO were set as follows. The two-value coefficient c_c was set to 0.01 and 10. The inertia weight of the primary particle γ was set to 0.3.

The sources of toolbox for other algorithm were give in Section 2.3. The population size of the GA, PSO, FEP and GSO was set to 50. The reproduction function of GA was conducted using uniform stochastic selection. The mutation rate, crossover method and crossover rate were set to default values. The parameters of PSO in equation (1.2.1) were set according to the research of [36], where the inertial weight was set to 0.73 and the learning factors were both set to 2.05. A population of 50 was used in the PSO algorithm.The tournament size in FEP was set to 10 for selection [9]. The tunings of BFAVP were discussed in Section 2.3.7.

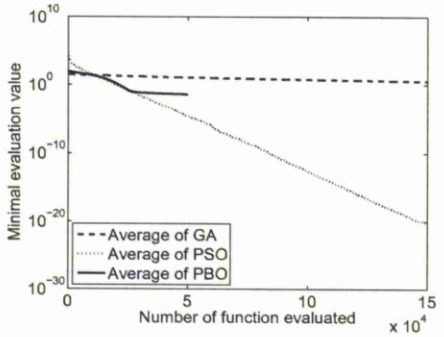
3.3.3 Experimental results

Table 3.3 shows the average minimal evaluation value obtained by GA, PSO, FEP, GSO, BFAVP, and PBO on benchmark functions $f_1 \sim f_5$ in 50 runs. It can be concluded that PSO shows better performance than other algorithms on functions f_1 and f_2. As discussed in Section 2.3.3, the decreasing inertial weight in PSO ensures that the search range of the entire population is reduced iteration by iteration. Although the major searching method in PBO is gradient searching, the small size of the population delays the convergence process. Meanwhile, the decreasing inertia weight provides PSO with more chance to stay around the global optima in unimodal functions that have only one optimum. These results have also been verified in Section 2.3.3: PSO is more suitable for solving high-dimensional unimodal functions. However, without the random factor, the diversity of population in PSO is limited. By exchanging the dimensional information in PBO, the algorithm is able to solve the problems, such as f_3 and f_5, which requires each dimension to be optimised simultaneously. The results also show that in most unimodal benchmark functions, PBO outperforms GA, and has a similar performance as BFAVP. Figures 3.2(a)~3.2(e) illustrate the convergence processes of GA, PSO, and PBO on high-dimensional unimodal functions, respectively.

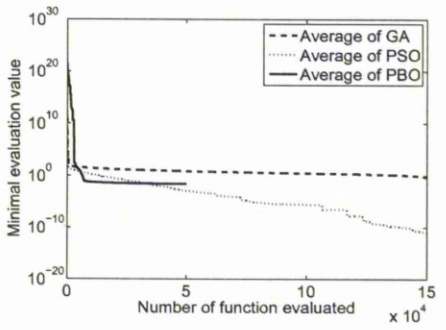
		GA*	PSO*	FEP*	GSO*	BFAVP	PBO
f_1	μ	3.1711	$\mathbf{3.6927 \times 10^{-37}}$	5.7×10^{-4}	1.9481×10^{-8}	9.9908×10^{-9}	2.2805×10^{-3}
	σ	1.6621	$\mathbf{2.4598 \times 10^{-36}}$	1.3×10^{-4}	1.9841×10^{-8}	8.7307×10^{-9}	4.8036×10^{-4}
f_2	μ	0.5771	$\mathbf{2.9168 \times 10^{-24}}$	8.1×10^{-3}	3.7039×10^{-5}	5.4701×10^{-5}	2.5380×10^{-2}
	σ	0.1306	$\mathbf{1.1362 \times 10^{-23}}$	7.7×10^{-4}	8.6185×10^{-5}	3.6527×10^{-5}	2.8340×10^{-3}
f_3	μ	7.9610	0.4123	0.3	0.1078	0.1504	$\mathbf{3.9315 \times 10^{-2}}$
	σ	1.5063	0.2500	0.5	3.9981×10^{-2}	0.1221	$\mathbf{4.4074 \times 10^{-3}}$
f_4	μ	338.5916	37.3582	5.06	49.8359	$\mathbf{7.7345 \times 10^{-4}}$	187.1295
	σ	361.497.	32.1436	5.87	30.1771	$\mathbf{2.10485 \times 10^{-3}}$	290.6316
f_5	μ	3.6970	0.1460	**0**	1.6000×10^{-2}	**0**	**0**
	σ	1.9517	0.4182	**0**	0.1333	**0**	**0**

$*$ The results are taken from [93] and [9].

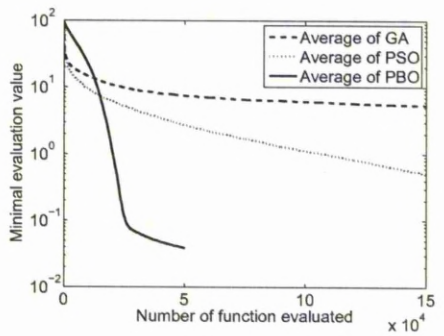
Table 3.3: Average best results of mean (μ) and standard deviation (σ) of GA, PSO, FEP, GSO, BFAVP and PBO on $f_1 \sim f_5$



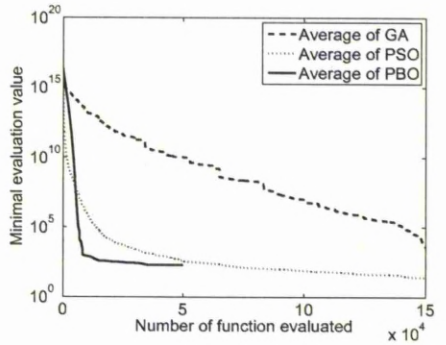
(a) f_1



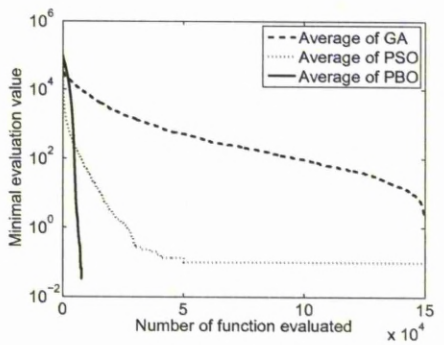
(b) f_2



(c) f_3



(d) f_4



(e) f_5

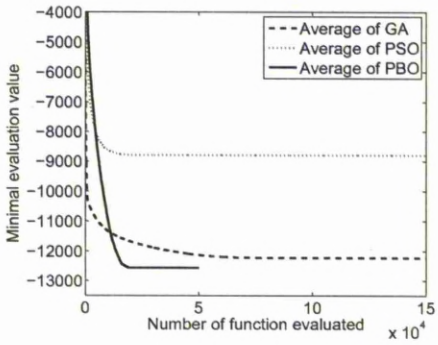
Figure 3.2: Convergence results of GA, PSO, and PBO in $f_1 \sim f_5$

As discussed in Section 2.3, with the increase of dimensions, the number of local optima in multimodal functions increases exponentially. Most EAs are unable to solve multimodal functions. Tables 3.4 and 3.5 list the average best results of the mean and standard deviation of GA, PSO, FEP, GSO, BFAVP, and PBO on multimodal benchmark functions. From these results, it can be found that Bacteria-inspired Algorithms (BIAs) have much better performance on this type of function. PBO converges with much less computation complexity because the random factor enhances the population diversity. Although PBO only has two bacteria, the pseudo individual always moves randomly in the search space. If the primary bacterium is trapped in local optimum, there is always a chance that the pseudo bacterium can attract it away from the area. In this experiment, PBO outperforms GA, PSO, FEP on all high-dimensional multimodal functions. Due to the mutation operator, the population diversity of GA is better than that of PSO. As a result, GA outperforms PSO with regards to several functions. GSO is claimed to have a superior performance on multimodal problem. In this experiment, PBO outperforms GSO on f_7 and f_9 due to the exchange of the position of optima on each dimension by the learning speed in equation (3.2.10).

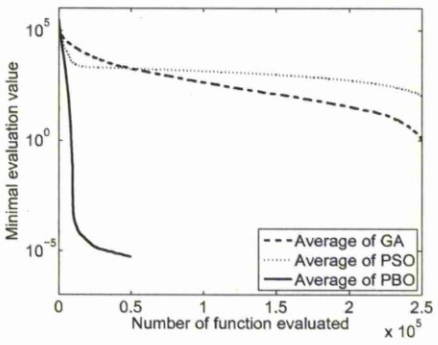
		GA*	PSO*	FEP*	GSO*	BFAVP	PBO
f_6	μ	-12566.0977	-9659.6993	-12554.5	-12569.4882	$\mathbf{-12569.4882}$	-12569.4833
	σ	2.1088	463.7825	52.6	2.2140×10^{-2}	1.3082×10^{-2}	$\mathbf{7.7188 \times 10^{-4}}$
f_7	μ	0.6509	20.7863	4.6×10^{-2}	1.0179	0.3319	$\mathbf{5.3881 \times 10^{-6}}$
	σ	0.2805	5.9400	1.2×10^{-2}	0.9509	0.3368	$\mathbf{2.0746 \times 10^{-6}}$
f_8	μ	0.8678	1.3404×10^{-3}	1.8×10^{-2}	2.6548×10^{-5}	$\mathbf{1.6325 \times 10^{-5}}$	1.4857×10^{-3}
	σ	0.2805	4.2388×10^{-2}	2.1×10^{-2}	3.0820×10^{-5}	$\mathbf{2.9589 \times 10^{-5}}$	1.8005×10^{-3}
f_9	μ	1.0038	1.0633×10^{-2}	2.6×10^{-2}	3.1283×10^{-2}	2.4649×10^{-2}	$\mathbf{6.1561 \times 10^{-3}}$
	σ	6.7545×10^{-2}	1.0895×10^{-2}	2.2×10^{-2}	2.8757×10^{-2}	4.2953×10^{-2}	$\mathbf{9.1762 \times 10^{-3}}$
f_{10}	μ	4.3572×10^{-2}	3.9503×10^{-2}	9.2×10^{-6}	$\mathbf{2.7648 \times 10^{-11}}$	1.9596×10^{-9}	1.9791×10^{-7}
	σ	5.0579×10^{-2}	9.1424×10^{-2}	6.1395×10^{-5}	$\mathbf{9.1674 \times 10^{-11}}$	2.0533×10^{-9}	4.1392×10^{-8}

$*$ The results are taken from [93] and [9].

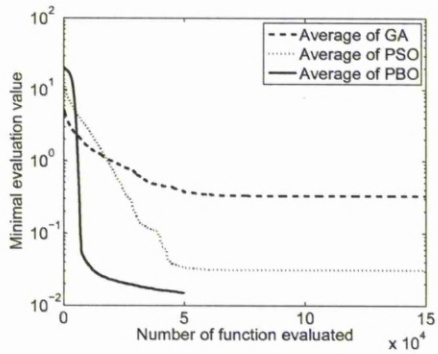
Table 3.4: Average best results of mean (μ) and standard deviation (σ) of GA, PSO, FEP, GSO, BFAVP, and PBO on $f_6 \sim f_{10}$



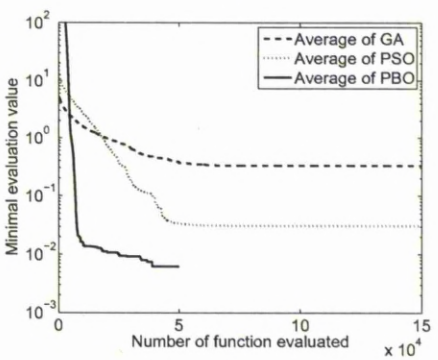
(a) f_6



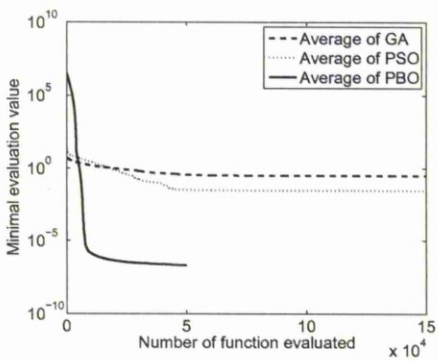
(b) f_7



(c) f_8



(d) f_9



(e) f_{10}

Figure 3.3: Convergence results of GA, PSO, and PBO in $f_6 \sim f_{10}$

		GA*	PSO*	FEP*	GSO*	BFAVP	PBO
f_{11}	μ	0.9989	1.0239	1.22	**0.9980**	**0.9980**	**0.9980**
	σ	4.4333×10^{-3}	0.1450	0.56	**0**	2.5080×10^{-16}	4.9535×10^{-13}
f_{12}	μ	7.0878×10^{-3}	3.8074×10^{-4}	5.0×10^{-4}	4.1687×10^{-4}	3.7594×10^{-4}	$\mathbf{3.6469 \times 10^{-4}}$
	σ	7.8549×10^{-3}	2.5094×10^{-4}	3.2×10^{-4}	3.1238×10^{-4}	4.2768×10^{-4}	$\mathbf{2.1342 \times 10^{-4}}$
f_{13}	μ	-1.0298	-1.0160	-1.03	$\mathbf{-1.0316}$	$\mathbf{-1.0316}$	$\mathbf{-1.0316}$
	σ	3.1314×10^{-3}	1.2786×10^{-2}	4.9×10^{-4}	**0**	5.3284×10^{-16}	1.2786×10^{-5}

$*$ The results are taken from [93] and [9].

Table 3.5: Average best results of mean (μ) and standard deviation (σ) of GA, PSO, FEP, GSO, BFAVP, and PBO on $f_{11} \sim f_{13}$

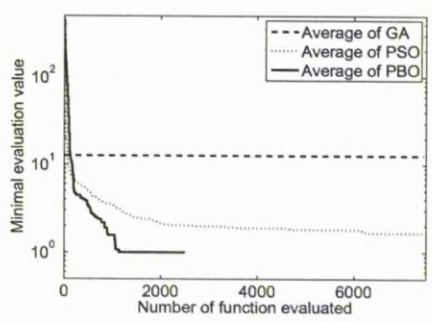
According to the results in Table 3.5, it can be seen that GSO, BFAVP, and PBO have obtained nearly the same results. The performance of PBO for low-dimensional multimodal functions is also improved. This merit is caused by the population size of PBO. Without wasting evaluations, it is easy for PBO to converge within a few iterations in low-dimensional functions. PBO can obtain a satisfactory solution when the number of evolutions is set to less as for other algorithms. Figures 3.3(a)~3.3(e) illustrate the convergence processes of GA, PSO, and PBO on high-dimensional multimodal functions, and Figures 3.4(a)~3.4(c) illustrate the processes on low-dimensional multimodal functions. In these figures, the number of function evaluations of PBO is set to one third of that of GA and PSO.

		GA*	PSO*	PBO
f_{14}	μ	6.9839×10^{-2}	4.3210×10^{-3}	$\mathbf{3.6987 \times 10^{-3}}$
	σ	2.8797×10^{-2}	1.4953×10^{-3}	$\mathbf{1.4890 \times 10^{-3}}$
f_{15}	μ	-9570.4365	-6898.9433	$\mathbf{-12569.4320}$
	σ	674.0944	791.74404	$\mathbf{1.4584 \times 10^{-2}}$
f_{16}	μ	3.0490	3.0030	**3.0016**
	σ	0.03503	1.5406×10^{-3}	$\mathbf{9.2863 \times 10^{-4}}$

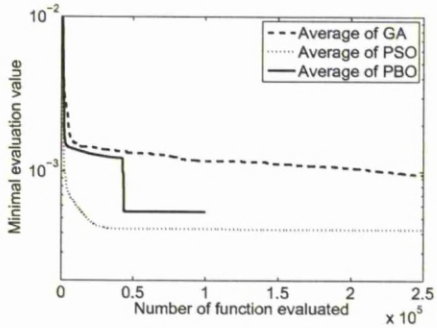
$*$ The results are taken from [93].

Table 3.6: Average best results of mean (μ) and standard deviation (σ) of GA, PSO, and PBO on $f_{14} \sim f_{16}$

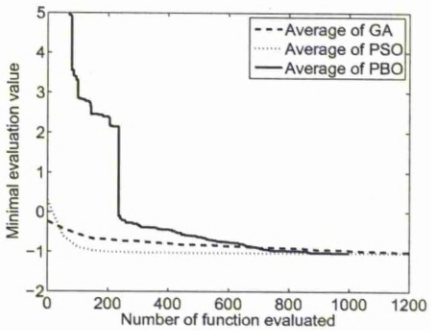
Functions $f_{14} \sim f_{16}$ were not evaluated in the research work of [93] and [9]. As a result, the performance of PBO in these functions with error interference were only compared with GA and PSO. Figures 3.4(d)~3.4(f) illustrate the convergence curves of GA, PSO, and PBO on dynamic functions. The results in Table 3.6 show that PBO outperforms GA and PSO on functions $f_{14} \sim f_{16}$, which have noise in the environment. PBO not only converges faster than the other two EAs, but also has a small standard variance, especially on dynamic unimodal functions. Compared with GA and PSO, PBO is more suitable to solve real-world dynamic optimisation problems.



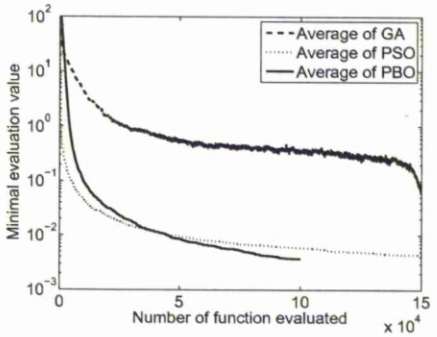
(a) f_{11}



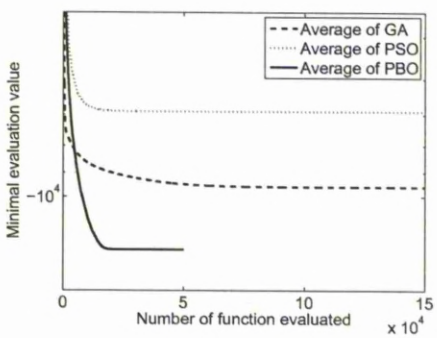
(b) f_{12}



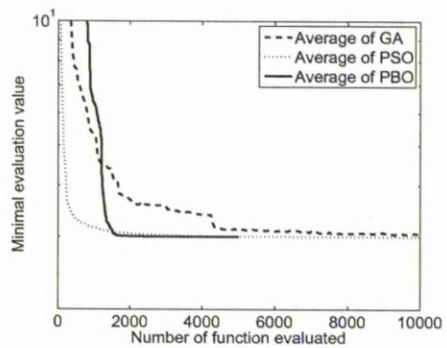
(c) f_{13}



(d) f_{14}



(e) f_{15}



(f) f_{16}

Figure 3.4: Convergence results of GA, PSO, and PBO in $f_{11} \sim f_{16}$

The major merit of PBO is its lower time consumption and higher efficiency. In order to analyse the computational complexity, the detailed mathematical and logical operations in each evaluation are listed in Table 3.7. In this table, it can be seen that with the same number of function evaluations, PBO saves 36% on the addition operations and 40% on the multiplication operations. This improvement greatly reduces time consumed, and makes PBO adapt more quickly to the dynamic environment. Figures 3.2(a)~3.4(f) give an illustration of the convergence capability of PBO. In summary, during the entire searching period, PBO is first in obtaining a stable solution in most cases, especially in multimodal functions.

Operators	PBO	PSO
Addition	4.5	7
Multiplication	6	10
Comparison	13	11
Random generation	2	2

Table 3.7: Mathematical and logical operators in PSO and PBO during each evaluation

3.4 Conclusion

In this chapter, a very simple optimisation algorithm, PBO is presented, which contains only two bacteria in the population. The pseudo gradient randomly places the pseudo bacterium along one dimension around the primary bacterium. The gradient-based searching method improves the accuracy of PBO. To increase the diversity of the algorithm, the pseudo bacterium may also be occasionally randomly placed far away from the primary bacterium. Meanwhile, to accelerate the convergence speed, simplified quorum sensing is introduced. This mechanism attracts bacterium to the passing location that has the best fitness value and the best partial gradient.

The experimental results have shown that this novel algorithm possesses a much better global search capability and convergence performance, given

the same or fewer number of function evaluations in comparison with GA and PSO.

Part II

Power System Applications Using Bacteria-inspired Optimisation Algorithms

Chapter 4

Applications of BFAVP to Power System

This chapter applies Bacterial Foraging Algorithm with Varying Population (BFAVP) to four power system optimisation problems: the Optimal Power Flow (OPF) problem, the partitioned OPF problem, the optimal allocation of FACTS devices, and the estimation of harmonic parameters. The performance of BFAVP is compared with two other Evolutionary Algorithms (EAs); Genetic Algorithm (GA) and Particle Swarm Optimiser (PSO). Experimental results show that BFAVP has superior performance in all of the applications studied.

4.1 Introduction

In power engineering, the power flow study is an important tool involving numerical analysis applied to a power system. A power flow study usually uses simplified notations such as a one-line diagram and per-unit system, and focuses on various forms of AC power rather than voltage and current. It analyses the power systems in normal steady-state operation. In addition to power flow study, OPF aims to find the conditions that have the lowest fuel cost per kilowatt generated.

Since OPF became an effective tool for dispatch planning, algorithms to

solve the OPF problem have been widely studied. The OPF problem has already been attempted as a static optimisation problem, by adopting conventional gradient-based methods. Recently, several EAs such as Genetic Algorithm (GA) and Particle Swarm Optimiser (PSO) have also been applied to this area. This chapter uses BFAVP to solve OPF problems and demonstrates the merits of BFAVP in power system applications.

OPF in large-scale power systems is limited by the computational complexity. This chapter presents a method to divide the IEEE 118-bus test case into two partitions. IEEE 118-bus test case represents a portion of the American Electric Power System. As there are 130 control variables in the test case, it is too complex to be solved by conventional EAs. In this partition method, the real and reactive power at the boundary of each partitions are fixed to constant values. By applying BFAVP to solve the OPF in each partition simultaneously, time consumption is greatly reduced. The experimental results are compared with GA and PSO to demonstrate the accuracy of BFAVP.

Moreover, a reactive power planning model is also presented in this chapter, which incorporates the FACTS devices. The locations of multi-type Flexible AC Transgression Systems (FACTS) devices and their control parameters are optimised by BFAVP to minimise the real power loss and to improve the voltage profile. The advantage of using BFAVP to determine the location of FACTS devices is demonstrated, in comparison with GA and PSO.

Apart from the power flow problem, this chapter also discusses the application of BFAVP to the estimation of harmonic parameters in a dynamic environments. Since the 1960s, the development of power electronic devices has expanded and the use of this equipment has multiplied in hundreds of applications in all industries. However, power electronic devices also feed harmonic currents into the power system, which is known as harmonic pollution. Harmonic pollution has serious hazards on electrical equipments, such as increasing the loss of transmission cables from harmonic current, reducing transmission capacity, increasing the motor loss and heat, limiting overload capacity and efficiency and lifetime, and even damaging the equipments [111]. With the

aim of reducing harmonic pollution in a power systems, a performance index based on the standard least square algorithm is adopted. Experimental results are presented to discuss the potential of the proposed approach in a dynamic environments.

The rest of this chapter is organised as follows. In Section 4.2, BFAVP proposed in Chapter 2 is applied to minimise the fuel cost for a power system. Then the simulation carried out on a partitioned OPF is studied in Section 4.3. To further reduce the power loss on transmission lines and fuel cost, four types of FACTS devices are introduced in Section 4.4. In Section 4.5, BFAVP is applied to estimate the harmonic parameters in dynamic environments. The conclusions about the applications of BFAVP are drawn in Section 4.6.

4.2 Optimal Power Flow Problem

4.2.1 Background

The OPF problem has been intensively studied as a network-constrained economic dispatch problem since its introduction by Carpentier [112] in 1962. The OPF problem aims to achieve the minimisation of fuel cost from a model of a power system by adjusting the control variables of the system, while satisfying a set of operational and physical constraints. As a result, the OPF problem can be formulated as a nonlinear constrained optimisation problem [113].

Various conventional optimisation methods have been developed to solve the OPF problem in the past a few decades, such as Nonlinear Programming NLP [4] [114], Quadratic Programming (QP) [115], Linear Programming (LP) [116], and interior point methods [117] [118]. However, these traditional methods do not guarantee finding the global optimum solution as discussed in Chapter 1. In most cases, the traditional methods are suitable for single-peak and linear objective functions. To tackle this problem, EAs, especially PSO, have been applied to solve the OPF problem [119]. These algorithms are all based on fixed-population evaluation, which limits their computational capability and introduces redundant computation into the optimisation process. Therefore,

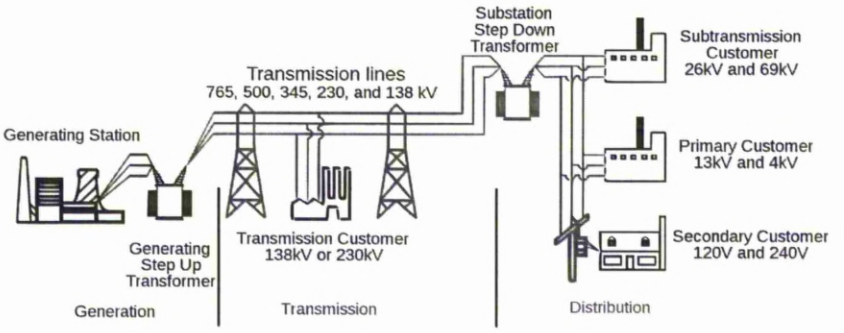


Figure 4.1: Diagram of an electrical system [3]

BFAVP is adopted to overcome these drawbacks.

A detailed OPF formulation is given in Section 4.2.2, including objective equations, constraint equations and inequality constraints. Then two conventional power system test cases, the IEEE 30-bus test case and IEEE 118-bus test case, are described, and the structures of the test cases are illustrated. Similar as IEEE 118-bus test case, IEEE 30-bus test case represents a portion of the American Electric Power System in the Midwestern US, which has 24 control variables. In Section 4.2.3, BFAVP was evaluated on a practical OPF problem, that focuses on minimising the fuel cost of systems. Finally, the experimental results are reported in this section to show the merits of the proposed algorithm in comparison with GA and PSO.

4.2.2 Formulations of optimal power flow

The diagram of an electrical power system [3] is illustrated in Figure 4.1. A typical power system consists of three parts: generating station generating electrical energy, transmission lines interconnecting the power system, and customers consuming the electrical energy.

For an OPF problem, the control variables include the generator real power, the generator bus voltages, the tap ratios of transformers and the reactive

power generation of volt-ampere reactive (var) sources [120]. For a generator, the real power output indicates the portion of power generated by plants that averaged over a complete cycle of the AC waveform. Meanwhile, reactive power output indicates the portion of power generated by plants returns to the source in each cycle. In the control variables, generator bus voltages indicates the voltage magnitude and angles of the bus with generator. Tap ratios indicates voltage ratio between the input and output of a transformer. Reactive power generation of var sources indicates the measure of reactive power an AC electric power generator.

The state variables are the slack bus power, the load bus voltage, the generator reactive power output, and the network power flows. In order to solve the power flow equations in each iteration of the OPF process, a slack bus is chosen by assuming that the slack bus power including voltage magnitude and voltage phase is known. Therefore, for each load bus, both the voltage magnitude and angle are unknown and can be solved. For generator buses, the voltage angle also can be solved.

The constraints of OPF problem include inequality constraints which are the limits of the control variables and state variables, and equality constraints which are the power flow equations.

The OPF problem can be formulated as a constrained optimisation problem as follows (nomenclature for power flow is listed in the Appendix B.3):

$$\min \quad F(u_{\mathrm{d}}, u_{\mathrm{c}}) \tag{4.2.1}$$

$$\text{s.t.} \quad G(u_{\mathrm{d}}, u_{\mathrm{c}}) \leq 0 \tag{4.2.2}$$

$$H(u_{\mathrm{d}}, u_{\mathrm{c}}) = 0 \tag{4.2.3}$$

where F is the optimisation objective function, G is a set of constraint inequalities, and H is a set of formulated constraint equalities. u_{d} is the vector of dependent variables such as the slack bus power P_{G_1}, the load bus voltage V_{L}, generator reactive power outputs Q_{G} and the apparent power flow S_k. In these variables, apparent power is the absolute value of complex power in the

power system. u_{d} can be expressed as:

$$u_{\mathrm{d}}^T = [P_{\mathrm{G}_1}, V_{\mathrm{L}_1}, ..., V_{\mathrm{L}_{N_{\mathrm{G}}}}, Q_{\mathrm{G}_1}, ..., Q_{\mathrm{G}_{N_{\mathrm{G}}}}, S_1, ..., S_{N_{\mathrm{E}}}]^T. \quad (4.2.4)$$

u_{c} is a set of the control variables such as the generator real power output P_{G} expected at the slack bus P_{G_1}, the generator voltages V_{G}, the transformer tap setting T, and the reactive power generations of the var source Q_{C}. Therefore, u_{c} can be expressed as:

$$u_{\mathrm{c}}^T = [P_{\mathrm{G}_2}, ..., P_{\mathrm{G}_{N_{\mathrm{G}}}}, V_{\mathrm{G}_1}, ..., V_{\mathrm{G}_{N_{\mathrm{G}}}}, T_1 ... T_{N_{\mathrm{T}}}, Q_{\mathrm{C}_1}, ..., Q_{\mathrm{C}_{N_{\mathrm{C}}}}]^T. \quad (4.2.5)$$

The equality constraints $H(u_{\mathrm{d}}, u_{\mathrm{c}})$ are the nonlinear power flow equations which are formulated as below:

$$0 = P_{\mathrm{G}_i} - P_{\mathrm{D}_i} - V_i \sum_{j \in N_i} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) \quad i \in N_0 \quad (4.2.6)$$

$$0 = Q_{\mathrm{G}_i} - Q_{\mathrm{D}_i} - V_i \sum_{j \in N_i} V_j (G_{ij} \sin\theta_{ij} + B_{ij} \cos\theta_{ij}) \quad i \in N_{\mathrm{PQ}}. \quad (4.2.7)$$

The inequality constraints $G(u_{\mathrm{d}}, u_{\mathrm{c}})$ are limits of control variables and state variables which can be formulated as:

$$\begin{aligned}
&P_{\mathrm{G}_i}^{\min} \le P_{\mathrm{G}_i} \le P_{\mathrm{G}_i}^{\max} \quad i \in N_{\mathrm{G}} \\
&Q_{\mathrm{G}_i}^{\min} \le Q_{\mathrm{G}_i} \le Q_{\mathrm{G}_i}^{\max} \quad i \in N_{\mathrm{G}} \\
&Q_{\mathrm{C}_i}^{\min} \le Q_{\mathrm{C}_i} \le Q_{\mathrm{C}_i}^{\max} \quad i \in N_{\mathrm{C}} \\
&T_k^{\min} \le T_k \le T_k^{\max} \quad k \in N_{\mathrm{T}} \\
&V_i^{\min} \le V_i \le V_i^{\max} \quad i \in N_{\mathrm{B}} \\
&|S_k| \le S_k^{\max} \quad k \in N_{\mathrm{E}}
\end{aligned} \quad (4.2.8)$$

The variables in equation (4.2.8) are the control variables in the power system. Control variables are chosen according to these inequalities and are used for solving the power flow equations in each iteration of the OPF process. Dependent variables in u_{d} are calculated based on the control variables by equations (4.2.6) and (4.2.7).

Moreover, the voltage V, reactive power Q, and apparent power flow S for all bus are also limited during the entire OPF process, which if formulated as

constrains of the objective function. To solve non-linear constrained optimisation problems, the most common method uses a penalty function to transform a constrained optimisation problem into an unconstrained one [121].

The objective function is generalised as follows:

$$F = F_{\text{cost}} + \sum_{i \in N_{\text{V}}^{\text{lim}}} \lambda_{V_i} (V_i - V_i^{\text{lim}})^2 + \sum_{i \in N_{\text{Q}}^{\text{lim}}} \lambda_{G_i} (Q_i - Q_{\text{G}_i}^{\text{lim}})^2 + \sum_{i \in N_{\text{E}}^{\text{lim}}} \lambda_{S_i} (|S_i| - S_i^{\max})^2 \tag{4.2.9}$$

where F_{cost} is the fuel cost of the system, λ_{V_i}, λ_{G_i}, and λ_{S_i} are the penalty factors. V_i^{lim} and $Q_{\text{G}_i}^{\text{lim}}$ are defined as:

$$V_i^{\text{lim}} = \begin{cases} V_i^{\max} & \text{if} \quad V_i > V_i^{\max} \\ V_i^{\min} & \text{if} \quad V_i < V_i^{\min} \end{cases} \tag{4.2.10}$$

$$Q_{\text{G}_i}^{\text{lim}} = \begin{cases} Q_{\text{G}_i}^{\max} & \text{if} \quad Q_{\text{G}_i} > Q_{\text{G}_i}^{\max} \\ Q_{\text{G}_i}^{\min} & \text{if} \quad Q_{\text{G}_i} < Q_{\text{G}_i}^{\min} \end{cases} \tag{4.2.11}$$

In order to improve power system stability, λ_{V_i}, λ_{G_i}, and λ_{S_i} are set to a large value. As a result, once V, Q, or S exceeds the limit, the trail solution will be eliminated from the population.

In summary, the decision variables of the OPF in this experiment are the generator real power output expected at the slack bus, the generator voltages, the transformer tap setting, and the reactive power generations of the var source. The objective of the optimisation is to reduce the total fuel cost of the power system and to restrict the voltage, reactive power, and apparent power flow within the bound for all buses.

In the following section, the IEEE 30-bus test case and IEEE 118-bus test case are adopted in the experimental studies. The IEEE 30-bus test case [4] represents a portion of the American Electric Power System (in the Midwestern US) as of December, 1961. There are 30 buses, 6 generators, and 40 branches in this model. Figure 4.2 illustrates the system layout of the IEEE 30-bus test case. The IEEE 118-bus test case represents a portion of the American Electric Power System (in the Midwestern US) as of December, 1962. There are 118

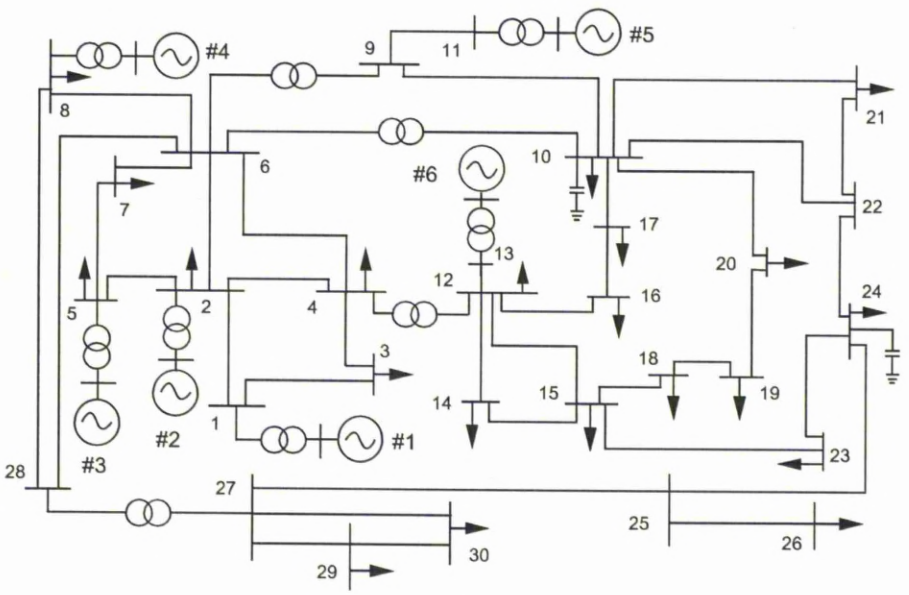


Figure 4.2: The layout of the IEEE 30-bus test case [4]

buses, 54 generators, and 186 branches in it. The number of control variables in the IEEE 118-bus test case is four times as much as the number in the IEEE 30-bus test case. Figure 4.3 illustrates the system layout of the IEEE 30-bus test case.

4.2.3 Experimental studies and results

In this experiment, BFAVP algorithm was tested on the standard IEEE 30-bus test case and IEEE 118-bus test case, which were adopted from [4]. In this case, the objective function is the total fuel cost of systems, which can be expressed as:

$$F_{\text{cost}} = \sum_{i=1}^{N_{\text{G}}} f_{\text{cost}_i} \tag{4.2.12}$$

$$f_{\text{cost}_i} = a_i + b_i P_{\text{G}_i} + c_i P_{\text{G}_i}^2 \tag{4.2.13}$$

where f_{cost_i} indicates the fuel cost with the unit of U.S. dollar per hour ($/h) of the i^{th} generator, a_i, b_i, and c_i are the fuel cost coefficients, and P_{G_i} is the

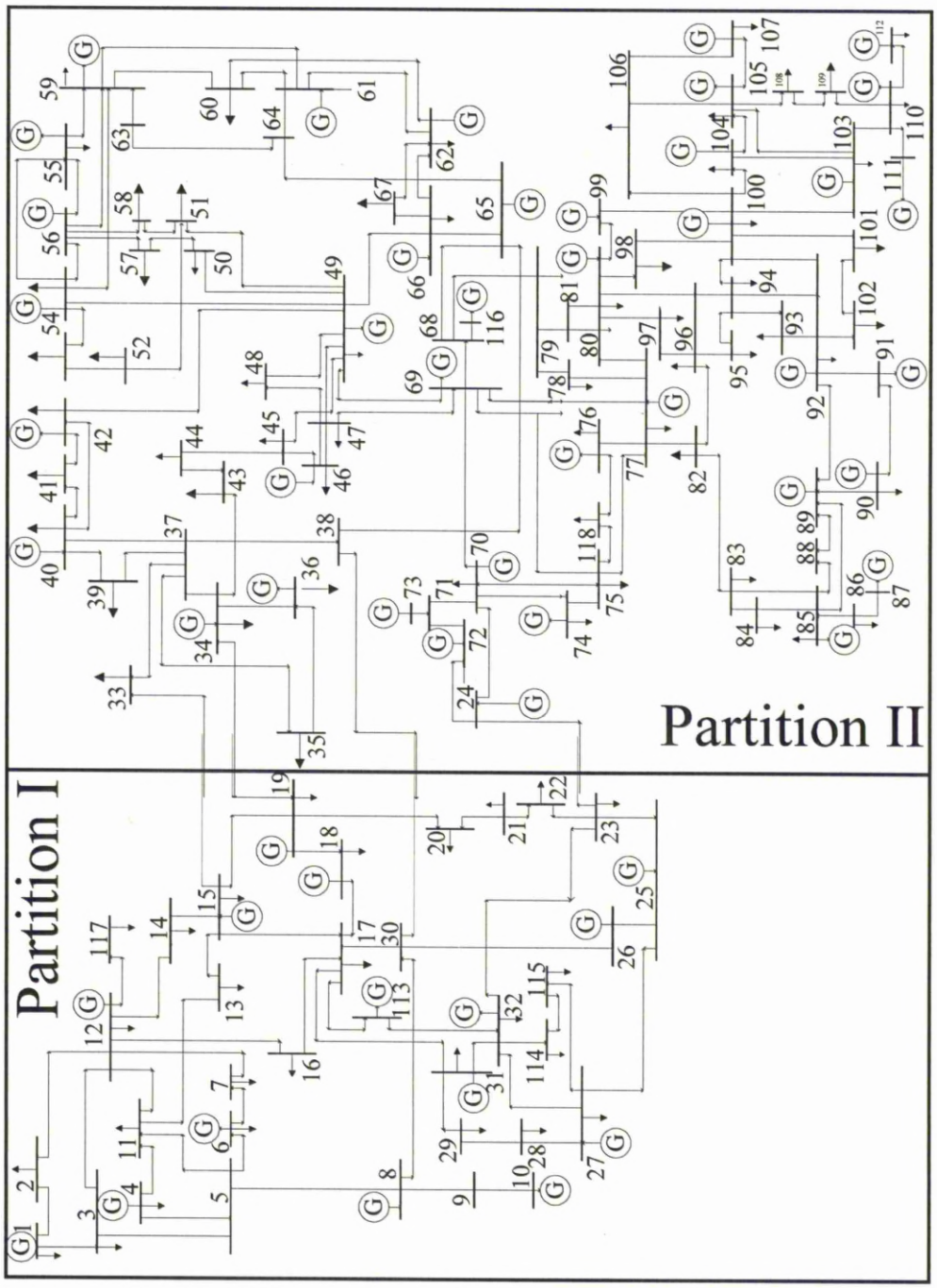


Figure 4.3: The layout of the IEEE 118-bus test case with two partitions [5]

real power output generated by the i^{th} generator.

In order to evaluate the proposed algorithm, the performance of BFAVP was compared with the GA and PSO in this section. The program and setting of GA were taken from [119], which demonstrated that GA is able to be applied to small and medium size power system. The implement of PSO for OPF was based on the program in [121]. A population of 50 individuals were used for GA and PSO. The number of power flow evaluations of these two algorithms was set to 1.5×10^3 for the IEEE 30-bus test case and 1.5×10^4 for the IEEE 118-bus test case. The initial population size of BFAVP was the same as that of PSO. However, during the evolutionary process, more individuals are reproduced in BFAVP. To prevent an unnecessary increase in the population size, the maximal population size was set to 150 for BFAVP. When BFAVP reached the same number of power flow evaluations as other two algorithms, the optimisation process was terminated.

For PSO, the inertia weight ω was set to 0.73, and the acceleration factors c_1 and c_2 were both set to 2.05, which follow the recommendations from [36]. For BFAVP, the average limit of an individual lifespan was 50 generations. The maximal length of a swim up along the gradient, N_c, was 4 steps.

The best result, average results and standard deviation of GA, PSO, and BFAVP from 30 runs are shown in Tables 4.1 and 4.2. GA fails to converge in the IEEE-118 bus test cases with a significantly large fuel cost due to the complexity of the evaluation function.

Algorithms	Best result	Average result	Standard deviation
GA	802.61	804.77	3.67
PSO	802.41	804.52	1.73
BFAVP	**802.13**	**803.05**	**0.94**

Table 4.1: Results from GA, PSO, and BFAVP on OPF for the IEEE 30-bus test case

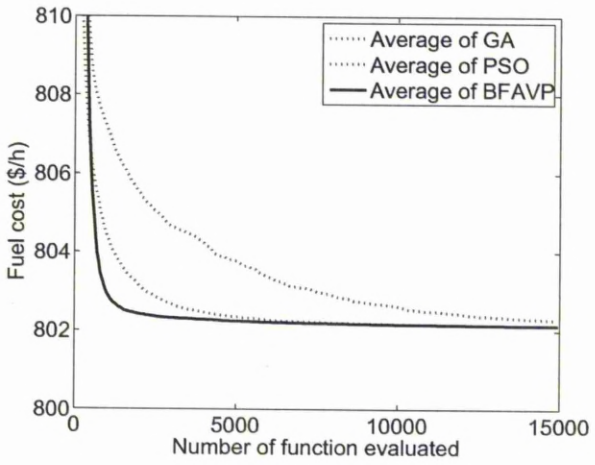


Figure 4.4: Convergence process of GA, PSO, and BFAVP on OPF

Algorithms	Best result	Average result	Standard deviation
GA	195146.2521	200716.6274	9783.0368
PSO	162383.5691	166795.5205	5907.8109
BFAVP	**143770.5205**	**144310.5205**	**5431.7914**

Table 4.2: Results from GA, PSO, and BFAVP on OPF for the IEEE 118-bus test case

Figure 4.4 illustrates the convergence process of GA, PSO, and BFAVP at the early stage in the IEEE 30-bus test cases. The horizontal axis represents the number of functions evaluated during the optimisation. The comparison was based on the same program running time in this figure. In this case, BFAVP converges faster than PSO and GA in the early period.

Figure 4.5 illustrates the population size of BFAVP during the OPF optimisation. The horizontal axis is the number of generations during the program's run time. The lag phase, logarithmic phase, and stationary phase can be seen in this figure. In the early period, bacteria obtain enough energy for reproduction, which leads to an increasing population size. Then the population size remains stable until the nutrition has been consumed. During the last stage,

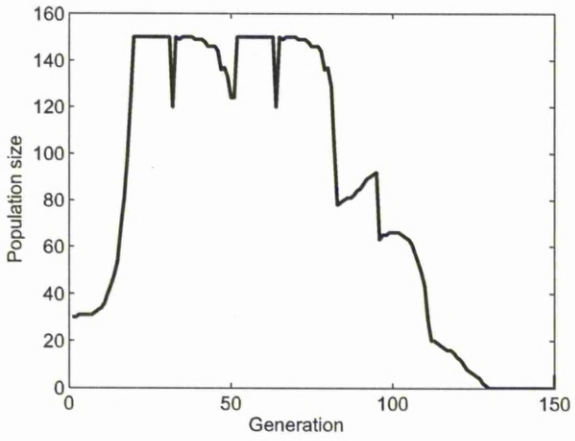


Figure 4.5: Varying population size of BFAVP during the OPF process

population size is reduced due to insufficient nutrition. The rugged deceasing population size in the death phase shows that the bacteria converge to several optima.

4.2.4 Summary

In this application, the experimental studies were carried out on the IEEE 30-bus and the IEEE 118-bus test cases to tackle the OPF problem, which aims to minimise the fuel cost. Although there are two algorithms proposed in this thesis, BFAVP is more suitable for applications that require a more accurate result. According to the results, BFAVP is able to obtain a lower fuel cost than PSO and GA. A more remarkable improvement can be spotted in terms of standard deviation. BFAVP demonstrated a satisfying performance with robustness. In terms of the convergence process, BFAVP is able to converge rapidly from the beginning of the evaluation process.

4.3 Partitioned Optimal Power Flow

4.3.1 Background

As an useful tool for the management of an interconnected transmission systems, OPF in large system intensively studied. Ideally, the OPF solution of a large, interconnected power system is obtained based on the assumption that the calculations will be commonly undertaken by an Independent System Operator (ISO) [122], which is able to access the entire network model and technical data. However, the centralised management of the interconnected transmission system is hindered by political and organisation issues, such as the difficulty in disclosing utility data, the lack of political intention to devolve power to a foreign centre, and large communication requirements between each node of the interconnected system and the central point for real-time data acquisition [123]. Moreover, the assemblage and maintenance of a huge database for the central management of the whole system or utility data results in huge CPU requirements and brings concerns about the reliability of the centralised operation and robustness of the centralised algorithm. Therefore, the centralised OPF solution for interconnected transmission systems is questionable with the current state of the art [124].

In contrast, a large interconnected power system can be geographically divided into a variety of regions, and each region can be operated by a different ISO. Each ISO is responsible for the management of its own regional transmission system, as well as cross-border trading with its neighbouring ISOs. As a result, computations can be performed in parallel for each region in the distributed multi-processor environment, which can potentially increase the available computation capacity and decrease the communication burden, allowing for faster and more reliable OPF solutions. In this section, a method to tackle a partitioned OPF problem is presented, which is suitable for application to very large interconnected transmission systems, and BFAVP is adopted to solve this problem. In the method, each ISO solves a partitioned OPF that includes its own service region and the borders it shares with other ISOs. The

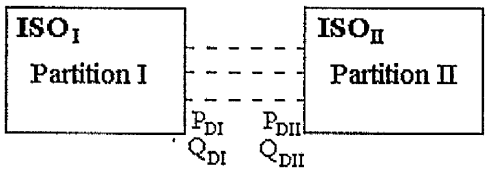


Figure 4.6: The connections of two partitions in a distributed power system [5]

partitioned OPF is similar to a standard OPF except that dummy generators are modelled at the border buses. The dummy generators mimic the effects of the external part of system and supply the region with real and reactive power. Therefore, solving the OPF in each region can be implemented by fast algorithms [125].

This section begins with an explanation about the experimental environment in the IEEE 118-bus case. In order to reduce time consumption, the test case is divided into two partitions. The layout and boundary constraints are given in Section 4.3.2. Then the BFAVP is applied to optimise the fuel cost for each of the partitions in Section 4.3.3. The computational time and fuel cost estimated by GA, PSO, and BFAVP are compared after the experiment.

4.3.2 Partitions in power systems

In the distributed power flow problems, the power system is separated into several partitions. Then a local reference bus is selected from the subsystems. The real and reactive power (P_D, Q_D) at both edges of the transmission lines between the subsystems are fixed to constant values. As a result, other subsystems connected to this one can be simulated as a fixed branch in the entire system. Figure 4.6 illustrates two partitions in a distributed power system.

In order to demonstrate the partitioned OPF experiment, the IEEE 118-bus test case is separated into two partitions, which are marked as Partition I and Partition II in Figure 4.3.

4.3.3 Experimental studies and results

In this section, the BFAVP algorithm was applied to the standard IEEE 118-bus test system, which was adopted from [4]. In order to meet the demands of separated power network managing, the system was separated into two independent partitions according to the geographical topology. Figure 4.3 illustrates the layout and the partition of the subsystem. Four transmission lines were chosen to be the exchange branches between each subsystem. The real and reactive power at both edges of the branch were fixed to a constant value. After the whole system was separated into two partitions, there were 34 control variables in Partition I and 97 control variables in partition II.

The decision variables and objective of the partition OPF in this experiment is same as those in OPF, which were discussed in Section 4.2.2. However, the power system are separated into partitions in this experiment, and each partition has a local fuel cost. In this case, the fuel cost of the j^{th} partition is expressed as:

$$F_{\text{cost}_j} = \sum_{i=1}^{N_{\text{G}_j}} f_{\text{cost}_{i,j}} \tag{4.3.1}$$

$$f_{\text{cost}_{i,j}} = a_{i,j} + b_{i,j} P_{\text{G}_{i,j}} + c_{i,j} P_{\text{G}_{i,m}}^2 \tag{4.3.2}$$

where $f_{\text{cost}_{i,j}}$ is the fuel cost (\$/h) of the ith generator in the j^{th} partition, $a_{i,j}$, $b_{i,j}$, and $c_{i,j}$ are fuel cost coefficients, $P_{\text{G}_{i_j}}$ is the real power output generated by the ith generator, and N_{G_j} is the number of generator buses in the partition.

For the purpose of algorithm performance comparison, GA [119] and PSO [35] were used in this section. The algorithm parameters were set to the same value as described in Section 4.2.3.

The best result, the average results and the standard deviations of GA, PSO, and BFAVP in partition I after 30 runs are shown in Table 5.1.

Algorithms	Best result	Average result	Standard deviation
GA	27977.1075	27951.3799	18.1547
PSO	27922.5952	27937.0599	8.8501
BFAVP	**27919.6098**	**27922.2982**	**1.2146**

Table 4.3: Best result, average best results, and standard deviations of GA, PSO, and BFAVP in partition I

Figure 4.7 illustrates the convergence processes of GA, PSO, and BFAVP. The horizontal axis represents the estimated time for the optimisation. *i.e.*, the comparison is based on the same program run time in this figure. In this case, BFAVP converges faster than PSO and GA in the early period. According to this figure, BFAVP outperforms GA and PSO in this high-dimensional multimodal application. This class of functions is difficult for most EAs because they are very likely to be trapped in local optima, due to the sole adoption of the gradient search method. The result is that the EAs following a pure gradient search principle are incapable of finding an accurate solution. Conversely, BFAVP models the quorum sensing behaviour, which randomly moves the bacteria from local optima to a better position. Accordingly, quorum sensing increases the probability of BFAVP finding the global optimum. As a result, with the control variables obtained by BFAVP, the system has the lowest fuel cost per hour compared to the other two algorithms.

Algorithms	Partition I	Partition II	Whole system
GA	27977.1075	174067.1635	195146.2521
PSO	27922.5952	138461.0163	162383.5691
BFAVP	**27919.6098**	**137501.1797**	**143770.5205**

Table 4.4: Average fuel cost in partitions and the whole system estimated by GA, PSO, and BFAVP

In order to compare the solution obtained by a distributed OPF with the whole system, these three algorithms were applied to optimise the control vari-

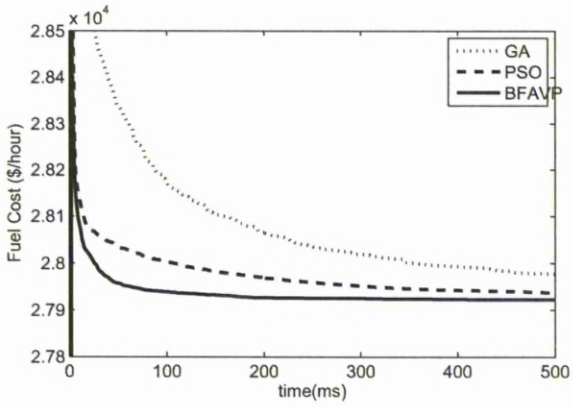


Figure 4.7: System fuel cost (Partition I) with the control variables obtained by GA, PSO, and BFAVP

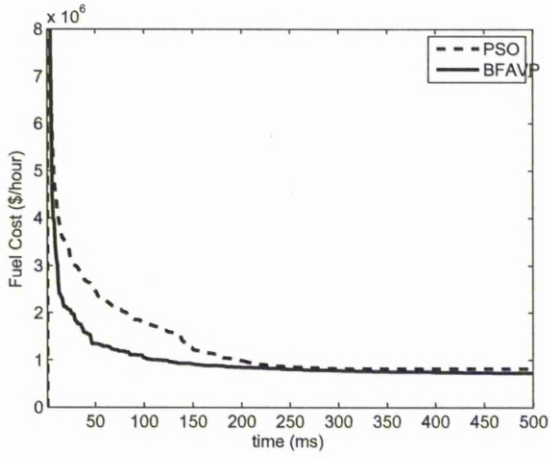


Figure 4.8: Whole system fuel cost with the control variables obtained by GA, PSO, and BFAVP

ables in partition II and the whole system. Figure 4.8 illustrates the convergence processes of PSO and BFAVP for the whole system. In this experiment, because the dimension of the solution increased from 34 to 130, GA failed to converge in the whole system optimisation. The average fuel cost in partitions and the whole system estimated by GA, PSO, and BFAVP are listed in Table 4.4. According to this table, as the system complexity increases, BFAVP obtains a more preferable solution compared to the other two algorithms.

Moreover, the result also shows that when the system is separated into partitions, some control variables are fixed to constant values. As a result, the flexibility of solution selection lessens compared to the conventional OPF. The fuel cost is also increased after the system separation, which can be expressed as:

$$F_{\text{cost}} < \sum_{i=1}^{N_{\text{P}}} F_{\text{cost}_j} \tag{4.3.3}$$

where F_{cost} is the fuel cost of the whole system, N_{P} is the number of partitions after the system separation, and F_{cost_j} is the fuel cost of the j^{th} partition.

4.3.4 Summary

In this section, BFAVP was adopted to solve partitioned OPF problems, which geographically divide the power network into a variety of regions and optimise the control variables in each region towards the target: minimising the cost of the network. In the experimental studies, the method was testified on a standard IEEE 118-bus test system. For the purpose of comparison, the optimisation algorithms of GA and PSO were also employed under the same simulation conditions. Experimental results have shown that BFAVP is suitable for solving the partitioned OPF problems and that it outperforms the other two algorithms.

4.4 Optimal Allocation of FACTS Devices

4.4.1 Background

In this section, BFAVP was applied to optimise the FACTS location in a power network. Modern power systems are facing new challenges due to deregulation and restructuring of electricity markets. The competition among utilities causes the increase of unplanned power exchanges. In concurrence with deregulation, electrical load continues to increase, and the transmission lines are running close to their thermal limits. Therefore, it is attractive for electric utilities to have a way of permitting a more efficient use of the transmission lines by controlling the power flows. FACTS devices represent a recent technological development in electric power systems, which allow utility companies to control power flow, increase transmission line stability limits, and improve the security of transmission systems [126]. In addition, FACTS devices can be used to maximise power transfer capability and minimise the power loss of the transmission systems, leading to an efficient utilisation of existing power systems [127]. In comparison with other corrective control strategies, such as generation rescheduling and load shedding, the utilisation of FACTS devices is a more economic alternative because it has a lower operational cost and introduces no additional cost [128].

However, due to the high installation cost, it is important to place FACTS devices optimally in a power network. The optimal location of FACTS devices has been attempted by using EAs such as GA [129] and PSO [130]. However, most of these studies have taken into account only the methods oriented towards technical criteria, *i.e.*, enhancement of system loadability and maximisation of the power system security margin [131], or methods that are directed at economical approaches, which are used to minimise the overall operation cost or total generation fuel cost. In fact, the optimal location of FACTS devices should be formulated as a multi-objective optimisation problem from both technical and economical points of view to make it capable of performing OPF and reactive power dispatch simultaneously. In this chapter, BFAVP is

applied to solve the multi-objective optimisation problem incorporated with a dominance method [132]. The location of FACTS devices, including their types and ratings, was optimised by applying BFAVP to minimise the fuel cost of all the generations, the installation cost of FACTS devices, and the power loss of transmission lines. Moreover, the system loadability was maximised within a security margin, *i.e.*, the thermal limits of the lines and voltage limits of the buses. Four types of FACTS devices, associated with their specific characteristics, were selected and modelled in this study and incorporated in the OPF problem.

The characteristic of each FACTS device is discussed in Section 4.4.2. Then BFAVP is implemented to meet the demands of multi-objective optimisation in Section 4.4.3. Experimental studies in Section 4.4.4 have been undertaken on the IEEE-14 bus test case and IEEE 30-bus test case. The results obtained are analysed to demonstrate the performance of BFAVP in the optimal placement of FACTS devices.

4.4.2 FACTS devices in power systems

FACTS device is a system composed of static equipment used for the AC transmission of electrical energy. By installing FACTS device, the controllability and power transfer capability can be improved. It is generally a power electronics-based system. Four types of FACTS devices are chosen to be installed to control power flow in this experiment: static var compensator (SVC), thyristor-controlled series capacitor (TCSC), thyristor-controlled phase shifting transformer (TCPST) and unified power flow controller (UPFC). The SVC makes it possible to enhance the functioning of a transmission network by increasing its loading margin [133]. It consists of a group of shunt-connected capacitor and reactor banks with fast control action by means of thyristor-based switching elements [134]. Hence, the SVC can be operated both for inductive and capacitive compensation. In the first case it absorbs reactive power while in the second the reactive power is injected. In this section, the mathematical models of FACTS devices are developed to perform the steady-state analysis.

Therefore, the SVC is modelled as an ideal reactive power injection as a shunt element which is incorporated into the sending-end of a transmission line as shown in Figure 4.9(a). The injected power of the SVC at bus i is denoted as:

$$\triangle Q_i = Q_{\text{SVC}} \tag{4.4.1}$$

The TCSC is a series type of FACTS devices. Through the firing angle control of thyristors, it is able to change the equivalent reactance of a transmission line to control power flow, improve system stability and increase the power transfer limit [135]. The ideal TCSC model is represented as a variant capacitive or inductive reactance method with resistance ignored. Because X represented the position of a individual in previous chapters, the reactance in this thesis is represented by E. Hence, the reactance of the transmission line, E_{L}, can be decreased or increased by controlling the TCSC directly. The mathematical model of the TCSC is shown in Figure 4.9(b), where the reactance of line ij is modified to:

$$E_{ij} = E_{\text{L}} + E_{\text{TCSC}}. \tag{4.4.2}$$

The TCPST is a phase-shifting transformer adjusted by thyristor switches to provide a rapidly variable phase angle. Phase shifting is obtained by adding a perpendicular voltage vector in series with a phase and the vector is derived from the other two phases via shunt-connected transformers. As a result, the voltage angle between the sending and receiving end of the transmission line can be regulated by the TCPST. In our study, the TCPST can be modelled as an ideal phase shifter which has series impedance equal to zero. It is inserted in series in a transmission line as shown in Figure 4.9(c) and may take a positive or negative value of angle, θ_{TCPST}. Hence, the voltage at bus i is adjusted to:

$$V_i^{'} = V_i \angle \theta_{\text{TCPST}}. \tag{4.4.3}$$

The UPFC is one of the most technically promising devices in the FACTS family [136]. It is able to control voltage magnitude and phase angle simultaneously, and can also independently provide reactive power injections. Therefore, the UPFC can provide voltage support and control real power flow. A

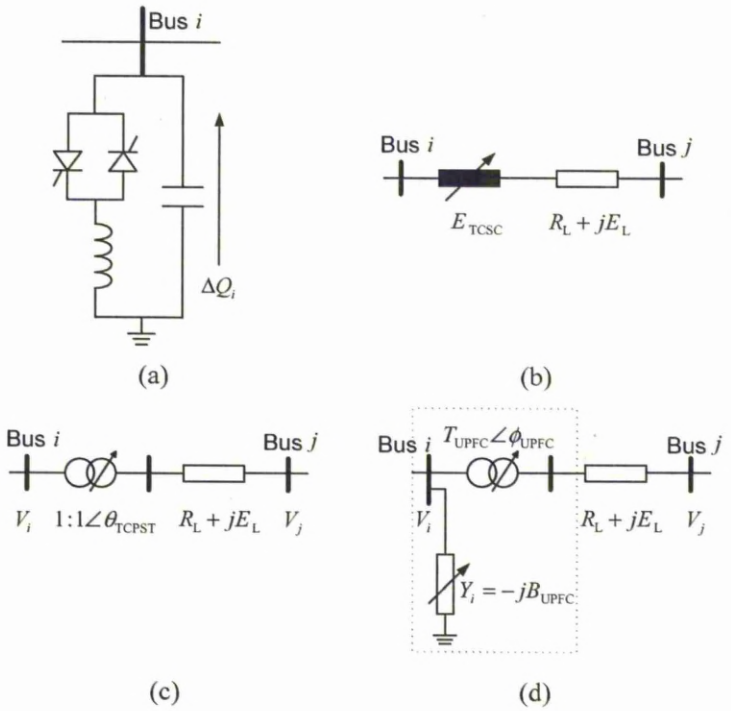


Figure 4.9: Models of FACTS devices. (a) SVC; (b) TCSC; (c) TCPST; (d) UPFC [6]

steady-state mathematical model for a UPFC is developed as shown in Figure 4.9(d). Since the UPFC conserves real power and generates or consumes reactive power, it can be modelled by an ideal transformer and a shunt branch. The turns ration and the variable shunt susceptance are independent variables, which are not directly associated with the UPFC input-output voltages and currents. The variables of the UPFC are defined as follows:

$$\begin{aligned} &T_{\text{UPFC}} && \text{transformer voltage magnitude turns ratio;} \\ &\phi_{\text{UPFC}} && \text{phase shifting angle;} \\ &B_{\text{UPFC}} && \text{shunt susceptance.} \end{aligned} \tag{4.4.4}$$

As seen in Figure 4.9(d), the voltage at bus i can be updated to:

$$V_i' = V_i T_{\text{UPFC}} \angle \phi_{\text{UPFC}}. \tag{4.4.5}$$

Allocation of FACTS devices can be formulated as a static nonlinear constrained optimisation problem, the solution of which determines the optimal location, type and rating of FACTS devices in a power network. It attempts to minimise one or more objective functions simultaneously while satisfying equality and inequality constraints of the network.

The cost of installation of FACTS devices is formulated as an objective function, which is to be minimised by optimal allocation of the FACTS devices in a power system network. The cost is estimated by:

$$C_{\text{F}} = 1000 \sum_i C_i S_i \tag{4.4.6}$$

where C_{F} is the total installation cost of FACTS devices in US$ and C_i is the cost of installation of each FACTS device in US$/KVar, and S_i is the operating range of the FACTS device in MVar. Each of S can be calculated respectively, by:

$$S = |Q_{\text{a}}| - |Q_{\text{b}}| \tag{4.4.7}$$

where Q_{a} is the reactive power flow (MVar) in the line after the installation of a FACTS device and Q_{b} is the reactive power flow in the line before installing a FACTS device.

For each type of FACTS device, the cost functions are different. The cost of installation of the SVC, TCSC and UPFC are taken from Siemens database and reported in [137] and [130]. They are given as:

$$\begin{aligned} C_{\text{SVC}} &= 0.0003S^2 - 0.3051S + 127.38 \\ C_{\text{TCSC}} &= 0.0015S^2 - 0.7130S + 15.75 \\ C_{\text{UPFC}} &= 0.0003S^2 - 0.2691S + 188.22 \end{aligned} \tag{4.4.8}$$

where C_{SVC}, C_{TCSC} and C_{UPFC} are installation costs for the SVC, TCSC and UPFC, respectively.

The cost of the TCPST is related to the operating voltage and current rating of the branch where the TCPST is allocated [138]. After the TCPST is installed, its cost C_{TCPST} is fixed and can be expressed as follows:

$$C_{\text{TCPST}} = dP_{\text{max}} + \text{IC} \tag{4.4.9}$$

where d is a positive constant representing the capital cost and IC is the installation cost of the TCPST. P_{max} is the thermal limit of the branch where the TCPST is to be installed. In this section, d and IC take the value of 10 and 9,000, respectively.

In order to make costs of FACTS devices comparable with generation fuel cost, the costs can be unified into US$ per hour as $/h. In this simulation, it supposes that the FACTS devices will be employed to regulate power flow for five years. Therefore, the average investment costs of installation of the FACTS devices are calculated as:

$$C_{\text{FH}} = \frac{C_{\text{F}}}{8760 \times 5} \quad (\$/\text{h}). \tag{4.4.10}$$

The objective, which aims to minimise the total generation fuel cost ($/h), is represented as:

$$F_{\text{cost}} = \sum_{i=1}^{N_{\text{G}}} f_{\text{cost}_i} = \sum_{i=1}^{N_{\text{G}}} (a_i + b_i P_{\text{G}_i} + c_i P_{\text{G}_i}^2) \tag{4.4.11}$$

where a_i, b_i and c_i are fuel cost coefficients, P_{G_i} is the active power output generated by the i^{th} generator, N_{G} is the total number of generators in the power network, and f_{cost_i} is the fuel cost for each generator.

The total real power loss in the transmission lines is represented as:

$$P_{\mathrm{L}} = \sum_{i=1}^{N_{\mathrm{E}}} P_i \tag{4.4.12}$$

where P_i the real power loss at line i, and N_{E} the number of transmission lines.

The reactive power injected or absorbed by the SVC is limited between:

$$-100 \text{ MVar} < \mathrm{Q_{SVC}} < 100 \text{ MVar}. \tag{4.4.13}$$

The value of the capacitance or inductance of the TCSC is limited to:

$$-0.8E_{\mathrm{L}} < E_{\mathrm{TCSC}} < 0.2E_{\mathrm{L}} \tag{4.4.14}$$

where E_{L} is the reactance of the transmission line at which the TCSC is placed. The angle of bus voltage, adjusted by the TCPST, is bounded by:

$$-5^\circ < \theta_{\mathrm{TCPST}} < 5^\circ. \tag{4.4.15}$$

The UPFC has three variables in which the transformer voltage magnitude turns ratio is chosen between the following range:

$$0.9 < T_{\mathrm{UPFC}} < 1.1 \tag{4.4.16}$$

and the phase shifting angle is constrained within:

$$-5^\circ < \phi_{\mathrm{UPFC}} < 5^\circ. \tag{4.4.17}$$

The shunt susceptance B_{UPFC} can be represented as injected or absorbed reactive power, Q_{UPFC}. Its working range is within:

$$-100 \text{ MVar} < \mathrm{Q_{UPFC}} < 100 \text{ MVar}. \tag{4.4.18}$$

The FACTS devices are placed in the network in order to increase the system loadability, and at the same time to prevent overloads and voltage violations. The system security constraint is based on indexes quantifying the system security state in terms of voltage levels and branch loading. Hence, the system security constraint contains two parts, and the formula of each

part in this experiment is adopted form [139]. The system security constraint can be represented as the sum of Vl and Bol, which indicates the violation of bus voltage and line flow limits considering all load buses and branches in the system. It is expressed as:

$$J_{\mathrm{SC}} = \sum_{i=1}^{N_{\mathrm{L}}} Vl_i + \sum_{j=1}^{N_{\mathrm{E}}} Bol_j \tag{4.4.19}$$

where N_{L} and N_{E} are the total numbers of load buses and transmission lines, respectively.

The first part of the system security constraint, Vl, concerns the voltage levels of each bus in the power network. The value of Vl_i for each bus is calculated as:

$$Vl_i = \begin{cases} 0 & V_{\mathrm{L}_i} \in [V_{\mathrm{L}_i}^{\min}, V_{\mathrm{L}_i}^{\max}] \\ \exp[\lambda_{\mathrm{r}}(|1 - V_{\mathrm{L}_i}| - 0.05)] - 1 & V_{\mathrm{L}_i} \notin [V_{\mathrm{L}_i}^{\min}, V_{\mathrm{L}_i}^{\max}] \end{cases} \tag{4.4.20}$$

where V_{L_i} is the voltage magnitude at bus i and λ_{r} represents the coefficient used to adjust the slope of the exponential function in the above equation. The equation indicates that appropriate voltage magnitudes are close to 1 per unit (p.u.). The value of Vl equals 0 if the voltage level falls between the voltage minimal and maximal limits. Outside the range, Vl increases exponentially with the voltage deviation.

The second part of the system security constraint, Bol, relates to the branch loading and penalty overloads in the lines. Similar to Vl, the value of Bol_j equals 0 if the j^{th} branch loading is less than its rating. Bol_j increases exponentially with the overload and it can be calculated from:

$$Bol_j = \begin{cases} 0 & S_j \leq S_j^{\max}, \\ \exp[(S_j^{\max} - S_j)/\lambda_{\mathrm{q}}] - 1 & S_j > S_j^{\max}, \end{cases} \tag{4.4.21}$$

where S_j and $S_j^{\max}$ are the apparent power in line j and the apparent power rate of line j, respectively. λ_{q} is the coefficient which is used to adjust the slope of the exponential function.

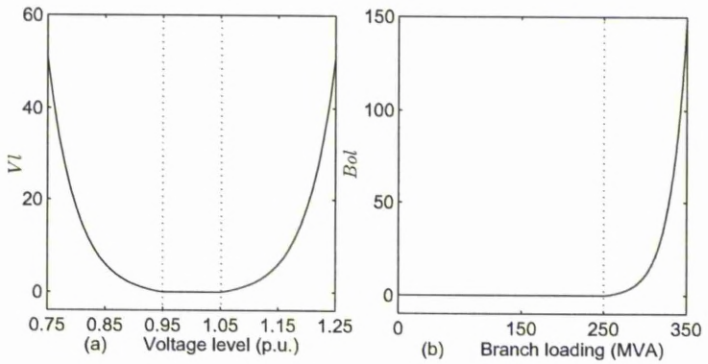


Figure 4.10: System security constraint. (a) Function Vl; (b) Function Bol

An illustration of functions Vl and Bol is given in Figure 4.10, in which $S_j^{\max}$ is 250 MVA, and $V_{Li}^{\min}$ and $V_{Li}^{\max}$ are 0.95 p.u. and 1.05 p.u., respectively. The coefficients λ_{r} and λ_{q} are both set to 20.

The power flow balance equation can be decoupled into real power load flow and reactive power load flow equations. The equations are constructed as equality constraints, given in Section 4.2.2. In these equations, P_{D_i} and Q_{D_i} are real and reactive power demands at bus i. In order to study the system loadability, a load parameter λ is included to reformulate the power flow equations. λ reflects the variation of power demands; thus, P_{D_i} and Q_{D_i} are modified as:

$$P_{\mathrm{D}_i}(\lambda) = \lambda P_{\mathrm{D}_i}, \tag{4.4.22}$$

$$Q_{\mathrm{D}_i}(\lambda) = \lambda Q_{\mathrm{D}_i}, \tag{4.4.23}$$

where $i = 1, \cdots, N_{\mathrm{D}}$, and N_{D} is the total number of power demand buses. $\lambda = 1$ indicates the base load case. Other constraints, including the limits of generator active power P_{G}, reactive power Q_{G} and voltage V_{G}, are represented as inequality constraints.

4.4.3 Implementation of algorithm

Initialisation

Initially, the system load parameter (SLP) was preset before the allocation of FACTS devices, *i.e.*, λ was set to 1.2, which means that the power demand at each load bus is increased by 20% from the base case value. The initial positions of bacteria X were generated randomly in such a way that the variables representing the position of each bacterium were in normalised form. The coordinates of X consisted of generator real power outputs P_G except the one at the slack bus P_{G_1} and generator voltages V_G.

The location, type and parameter settings of FACTS devices were control variables of X as well. If the number of FACTS devices to be installed was not known, it was supposed that there is an equal possibility of each branch in a power network having one single type FACTS device placed. N_{E} was the total number of branches in the network; thus, N_{E} variables in X took binary values of 0 or 1. The value '1' denotes that a FACTS device was installed at the corresponding branch, while '0' means that none were placed at the branch.

The type of the FACTS device at each branch was also denoted as control variables of X. N_{E} number of variables in X took integral values between 1 to 4, which indicates which type of FACTS device is installed at the corresponding branch. Here, value '1' was used to indicate an SVC, '2' for a TCSC, '3' for a TCPST and '4' for an UPFC. Moreover, the parameter settings for each type of FACTS device were all included as variables in X. N_{E} variables were required to denote the settings of the SVC, TCSC and TCPST, respectively. For the UPFC, $3N_{\mathrm{E}}$ variables were used to set the three variables of each UPFC. In total, each position of bacterium X has $8N_{\mathrm{E}}$ variables: N_{E} locations, N_{E} types, and $6N_{\mathrm{E}}$ settings.

Before evaluation of objective functions, the settings of FACTS devices and the system load factor were implemented into a power flow equation. For each solution X, the settings of each FACT device were multiplied by the variable representing its location. As a result, if a FACTS device was placed at the

branch ('1'), the setting of the FACTS device took the values given; thus, the corresponding parameters of the branch were revised. In contrast, if no FACTS device was installed at the branch ('0'), the setting of the FACTS device turned back to zero, which means that it has no impact on the parameters of the branch.

The above discussion concerns optimal allocation of FACTS devices with the SLP previously given. The objective is trying to find how many and what type of FACTS devices are required to be installed. However, if the number of FACTS devices is given and the objective is to find out the maximal system loadability with the help of these FACTS devices, the variables representing the location, type and settings of FACTS devices are different from the previous scenario. In this situation, N_{F} was used for integral variables to denote the location of FACTS devices. Each variable took an integer from $[1, N_{\mathrm{E}}]$, which represents the ordinal number of the branch where the FACTS device is located. If more than one FACTS device was to be installed, it was necessary to verify that only one device is placed in a branch. If two FACTS devices were placed in the same branch, one of them was removed from that branch and placed in some other branch where no FACTS device was installed. Another N_{F} number of variables was required to indicate the type of FACTS devices. As discussed previously, these variables had integral values from 1 to 4, which denote the SVC, TCSC, TCPST and UPFC, respectively. The variables used to represent the settings of FACTS devices were given in a similar manner. In total, the position of each bacterium consisted of $6N_{\mathrm{F}}$ variables, which were initialised to denote the settings of all the FACTS devices to be installed (N_{F} settings for the SVC, TCSC and TCPST, respectively; $3N_{\mathrm{F}}$ for the UPFC).

Evaluation of objective functions

The formation of evaluation values depends strongly on the optimisation goals of the power system operators and the available control variables. As indicated, the goal of optimisation for the first scenario is the optimal placement of FACTS devices into a power network in order to minimise overall opera-

tion cost and power loss within the security margin. Therefore, the presented problem becomes a multi-objective optimisation one that has two different evaluation values to be minimised simultaneously, which can be denoted as:

$$F(X) = [f_1(X), \ f_2(X)] \tag{4.4.24}$$

where F is known as the objective vector, f_1 and f_2 are the two objective functions to be optimised.

The first objective function represents the overall operation cost, including the total generation fuel cost ($/h) and the installation cost of FACTS devices ($/h). It is expressed as:

$$f_1(X) = F_{\text{cost}} + C_{\text{FH}}. \tag{4.4.25}$$

The second objective function calculates the total real power loss in transmission lines. The system security constraint is considered in the function by multiplying a penalty factor ω. As a result, the second function $f_2(X)$ is converted to an unconstrained optimisation problem and expressed as:

$$f_2(X) = P_{\text{L}} + \omega J_{\text{SC}}. \tag{4.4.26}$$

For the second scenario in which the number of FACTS devices N_{F} is given priorly, the goal of optimisation is the optimal placement of FACTS devices into a power network in order to minimise overall operation cost and increase system loadability within the security margin. Hence, the second objective function can be proposed as the maximisation of the SLP λ within the security margin, which can be presented as:

$$f_2(X) = \frac{1}{\lambda} + \omega J_{\text{SC}}. \tag{4.4.27}$$

The evaluation values can be estimated dependently and individually, and the selection depends upon the requirements of the power system operator. If only one of the objective functions is to be optimised, the problem becomes a single objective optimisation problem.

In the optimisation process, the parameter of the branch or bus is updated according to the settings of the FACTS devices installed and the value of the

SLP for each solution. Power flow is calculated using the Newton-Raphson method, and then line flows and voltage at buses are obtained. Using these values, the objective functions for each X are estimated.

Pareto-optimal solution and best compromising solution

Multi-objective optimisation is the process of simultaneously optimising two or more conflicting objectives subject to certain constraints. For single-objective optimisation problems, the goal is to find the global maximum or minimum subject to the objective function. In contrast, multi-objective optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. The optimisation process aims to locate a series of solutions which would decrease some fitness without causing a simultaneous increase in at least one of the other fitness.

In a typical multi-objective optimisation problem, there exists a set of solutions that are superior to the rest of the solutions in the search space when all objectives are considered but are inferior to other solutions in the space in one or more objectives. These solutions are known as Pareto-optimal solutions or non-dominated solutions [140]. In order to determine whether a solution is a Pareto solution, the concepts of dominance relation and Pareto-front are introduced.

The multi-objective optimisation problem can be formulated as follows:

$$F(X) = [f_1(X), f_2(X), \cdots, f_{m_\mathrm{f}}(X)] \tag{4.4.28}$$

subject to:

$$G_i(X) \leq 0 \; i = 1, 2, \cdots, m_\mathrm{g} \tag{4.4.29}$$

$$H_i(X) = 0 \; i = 1, 2, \cdots, m_\mathrm{h} \tag{4.4.30}$$

where $X = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$ is the vector of the parameters to be optimised, $f_i(i = 1, \cdots, m_\mathrm{f})$ are the objective functions, and $g_i(i = 1, \cdots, m_\mathrm{g}), h_i(i = 1, \cdots, m_\mathrm{h})$ are the constraint functions of the problem.

Given any two vectors $X, Y \in \mathbb{R}^n$, where $X = (x_1, x_2, ...x_n)$, $Y = (y_1, y_2, ...y_n)$, and $X \neq Y$, if $x_i \leq y_i \; \forall \; i = 1, \cdots, n$, then vector X dominates vector Y [141].

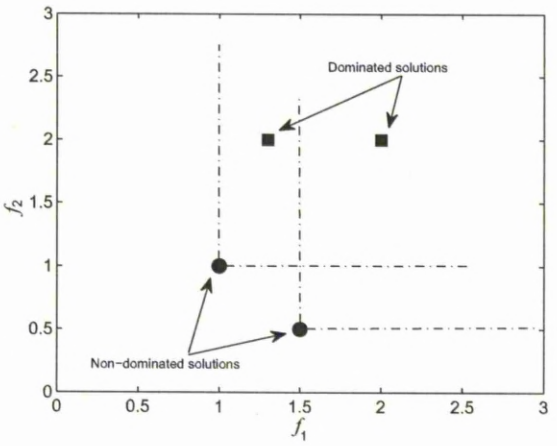


Figure 4.11: Dominance relation in multi-objective problems

The solutions, which are not dominated by any other solution, are called non-dominated solution. Figure 4.11 illustrates an example of the non-dominated relation and dominated solution.

Given a set of alternative solutions for a set of individuals, a change from one allocation to another that can make at least one individual better off without making any other individual worse off is called a Pareto-improvement. An allocation is defined as Pareto-optimal when no further Pareto improvements can be made. The set of non-dominated solutions consists of a Pareto-optimal solution set, which is expressed as:

$$\mathcal{P} = \{X \in \mathcal{F} | X \text{ is the Pareto optima}\} \tag{4.4.31}$$

where $\mathcal{F} \subset \mathbb{R}^n$ is the feasible region. From the Pareto-optimal solution set, the Pareto-front can be generated for a multi-objective decision. The Pareto-front $\mathcal{F}_{\mathrm{P}}$ is defined by:

$$\mathcal{F}_{\mathrm{P}} = \{F(X) \in \mathbb{R}^n | X \in \mathcal{P}\}. \tag{4.4.32}$$

For each solution, there must exist a domination relation between the solution considered and other solutions. Given the positions of two bacteria X_j and X_k, if $f_1(X_j) \leq f_1(X_k)$ and $f_2(X_j) \leq f_2(X_k)$, then the solution X_j dominates the solution X_k. In other words, X_j is a non-dominated solution, and

X_k is a dominated solution. The set of non-dominated solutions constitutes the Pareto-optimal solution set $\mathcal{P}^*$.

The goal of BFAVP for multi-objective optimisation is not only to guide the search towards the Pareto-front but also to maintain population diversity in the set of non-dominated solutions. In the application of BFAVP to multi-objective optimisation, the concept of dominance is used to find non-dominated individuals, which are stored externally. Firstly, an empty external Pareto-optimal solution set is created when an initial population of bacteria are generated with random positions. The evaluation values of each bacterium are then calculated as described previously. After the evaluation of the objective functions, the non-dominated individuals can be found, and their evaluation values are copied into the external Pareto-optimal solution set. Therefore, all dominated solutions are removed from the Pareto-optimal solution set.

For each bacterium, a random tumbling angle is generated and the bacterium moves towards a new position. The evaluation values for the bacterium at the new position are calculated. If either evaluation value is improved, the bacterium will continue to move in the same direction until it reaches the maximal step limit. The external Pareto-optimal solution set is updated each time when the bacterium moves to a new position. Because individuals in the external Pareto-optimal solution set indicate a set of the best solutions, each bacterium randomly selects an individual in the external Pareto-optimal solution set as the direction towards which to move. In order to enhance the population diversity, mutation is applied to the remaining individuals. The procedures are repeated until the maximum number of iterations is reached.

After the maximum number of interactions is reached, the solutions stored in the external Pareto-optimal solution set compose the final Pareto-optimal solution set. As each solution in the Pareto-optimal solution set has the same importance for the decision, it is hard to measure whether the power system is improved after installing FACT devices. Afterwards, the best compromising solution is introduced to be a reference solution in each Pareto-optimal solution set. Best compromising solution can be chosen from the Pareto-optimal

solution set based on a decision maker. For each non-dominated solution X_k in the Pareto-optimal solution set, the normalised membership function μ^k is calculated. The individual X_k with the maximal value μ^k is considered the best compromising solution.

The best compromising solution from among the entire Pareto-optimal solution set can be chosen based on a decision maker. The i^{th} objective function $f_i(X)$ is represented by a membership function μ_i, which is defined as:

$$\mu_i = \begin{cases} 1 & f_i = f_i^{\min} \\ \frac{f_i^{\max} - f_i}{f_i^{\max} - f_i^{\min}} & f_i^{\min} < f_i < f_i^{\max} \\ 0 & f_i = f_i^{\max} \end{cases}, \tag{4.4.33}$$

where $f_i^{\min}$ and $f_i^{\max}$ are the minimal and maximal value of the i^{th} objective function among all non-dominated solutions in the Pareto-optimal solution set, respectively. For each non-dominated solution x_k, the normalised membership function μ^k is calculated as:

$$\mu^k = \frac{\sum_{i=1}^{N} \mu_i^k}{\sum_{k=1}^{Z} \sum_{i=1}^{N} \mu_i^k}, \tag{4.4.34}$$

where N and Z are the number of objective functions, and non-dominated solutions, respectively. The individual x_k with the maximal value μ^k is considered the best compromising solution. As finite solutions can be found by BFAVP, there always exists a best compromising solution that can be selected as the reference solution.

4.4.4 Experimental studies and results

Single objective case studies

In order to verify its feasibility, BFAVP was applied for optimal placement of multi-type FACTS devices on the IEEE 14-bus test case, which represents a portion of the American Electric Power System (in the Midwestern US) as

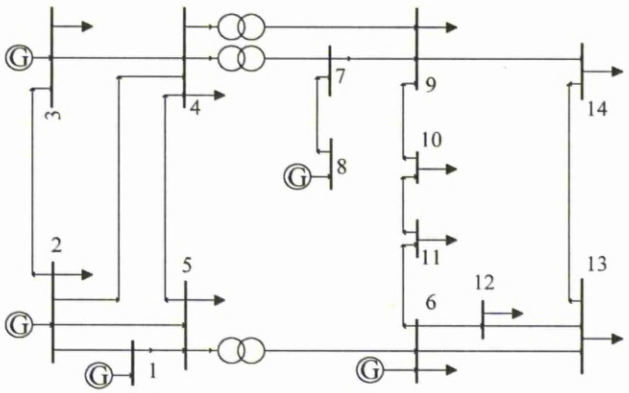


Figure 4.12: Single-line diagram of the IEEE 14-bus test case [7]

of February, 1962 [7]. The system has five generators at buses 1, 2, 3, 6 and 8. The voltage magnitude limits of all buses were set to 0.94 p.u. for lower bound and to 1.06 p.u. for upper bound. The single-line diagram of the IEEE test system is shown in Figure 4.12. Experimental studies were undertaken in order to obtain optimal solutions and provide solution for the placement of FACTS devices. In the implementation of BFAVP to allocate multi-type FACTS devices, the population size and the maximum number of generations were selected as 50 and 300, respectively. The percentages of the bacteria on selection of quorum sensing and mutation were set to be 80% and 20%, respectively. The maximal length of swimming up along the gradient for each bacterium, n_c, was set to 4 steps.

To decrease overall operation costs, first of all, suppose that FACTS devices had not been installed on the IEEE 14-bus test case and the SLP λ was fixed at $\lambda = 1.8$. Under this circumstance, the total generation fuel cost C_{FC} was minimised as a single objective optimisation problem by BFAVP. For the purpose of comparison, two population-based EAs, GA and PSO, were applied to the optimisation of the fuel cost as well. The best results for fuel cost over 20 runs are given in the left column of Table 4.5, which show that BFAVP gets a better optimisation result than GA and PSO.

In order to enhance the operating efficiency of the system, four types of

FACTS devices, *i.e.*, the SVC, TCSC, TCPST and UPFC, were installed and located at different branches of the network. The overall operation cost f_1, including the total fuel cost C_{FC} and the installation cost of FACTS devices C_{FH}, was optimised by the three optimisation algorithms, respectively. The best results for the fuel cost found by the three algorithms are given in the right column of Table 4.5.

Algorithms	Without FACTS (\$/h)		With FACTS (\$/h)		
	f_1	C_{FC}	f_1	C_{FC}	C_{FH}
GA	17698.4	17698.4	16770.1	16729.5	40.5899
PSO	16622.5	16622.5	16611.4	16604.2	7.17208
BFAVP	**16614.4**	**16614.4**	**16594.0**	**16592.2**	**1.85707**

Table 4.5: Optimisation of the overall operation cost for the IEEE 14-bus test case

As seen from Table 4.5, BFAVP achieved the best result in the case of placing multi-type FACTS devices in the IEEE 14-bus test case. Compared with the case that no FACTS device was installed in the IEEE 14-bus test case, the total fuel cost has been decreased from 16614.4 to 16592.2 \$/h. Even if the installation cost of FACTS devices is included, the overall operation cost is 16594.0 \$/h, which is less than the fuel cost of the system without FACTS devices installed. It indicates that the optimal allocation of FACTS devices helps to decrease overall operation cost.

The optimal allocation of the FACTS devices found by BFAVP, including their location and parameters, are listed in Table 4.6. In total, ten FACTS devices (1 SVC, 1 TCSC, 7 TCPST and 1 UPFC) are placed at different branches to improve the power flow in the network. The average investment cost is 1.85707 \$/h for the installation of these FACTS devices, and the total installation cost is US\$ 81386 as given in Table 4.6.

FACTS devices	Location (bus)		Device setting		Cost C_F ($)
	From	To			
SVC	6	11	Q_{SVC}:	25.366	
TCSC	6	13	X_{TCSC} :	-0.0265	
TCPST	1	2	θ_{TCPST}:	2.1821	
	3	4		-1.9675	
	4	5		-0.0198	
	4	9		3.1601	8.1386×10^4
	5	6		1.8707	
	9	10		0.3508	
	13	14		0.3138	
UPFC	7	9	T_{UPFC}:	1.0371	
			ϕ_{UPFC}:	-1.8108	
			Q_{UPFC}:	35.270	

Table 4.6: Test results of BFAVP for optimal location and parameter settings of FACTS devices in the IEEE-14 bus test case

In order to maximise the loadability of the power system, the SLP λ was added as another control variable in X. The value of λ was limited in [1, 3], which represents a 200% increment of power demands was allowed in optimisation process. The evaluation value of $f_2(X)$ given in equation (4.4.27) was optimised by BFAVP as a single objective, and then the location, setting of FACTS devices and optimal installation cost were obtained. To study the effect of each type of FACTS devices on maximising the system loadability, the IEEE 14-bus test case was installed with single and multi-type FACTS devices, respectively. The detailed results, including the maximal SLP, the location and setting of FACTS devices, and the cost of each type of device are given in Table 4.7.

Case	SLP λ	FACTS devices	Location From	Location To	Device setting	Cost C_F ($)
Single type	2.1023	SVC	2	3	Q_{SVC} : 67.743	1.6377×10^6
			7	8	8.2932	
			9	14	18.554	
	1.8547	TCSC	1	2	X_{TCSC} : -0.0314	7.9914×10^5
			4	5	-0.0204	
			9	14	-0.0972	
			13	14	-0.0985	
	1.7812	TCPST	6	13	θ_{TCPST} : -0.0589	2.3000×10^5
			7	8	2.9391	
	1.9226	UPFC	1	2	T_{UPFC} : 0.9509	2.5402×10^5
					ϕ_{UPFC}: 0.2951	
					Q_{UPFC}: 34.764	
			3	4	T_{UPFC} : 0.9808	
					ϕ_{UPFC}: -2.9960	
					Q_{UPFC}: 53.178	
			9	14	T_{UPFC} : 1.0052	
					ϕ_{UPFC}: -3.9619	
					Q_{UPFC}: 29.110	
Multi -type	2.5334	SVC	4	7	Q_{SVC} : 25.631	3.0676×10^6
		TCPST	2	3	θ_{TCPST}: -0.2914	
			6	11	1.0153	
		UPFC	3	4	T_{UPFC}: 0.9510	
					ϕ_{UPFC}: 0.5455	
					Q_{UPFC}: 87.176	
			4	9	T_{UPFC}: 1.0935	
					ϕ_{UPFC}: -1.7265	
					Q_{UPFC}: 24.522	
			7	9	T_{UPFC}: 0.9643	
					ϕ_{UPFC}: -3.2152	
					Q_{UPFC}: 28.729	
			9	14	T_{UPFC}: 0.9517	
					ϕ_{UPFC}: 0.4296	
					Q_{UPFC}: 57.164	

Table 4.7: Optimal allocation of FACTS devices in the IEEE-14 bus test case to increase the system loadability

To minimise the real power loss, suppose the load factor was still fixed at 1.8 for the IEEE 14-bus test case. The total real power loss of the system was calculated without installing FACTS devices and the result is 19.4121 MW. Afterwards, to improve the performance of reactive power planning of the system, multi-type FACTS devices were installed and each of them is located at a different branch. The evaluation value $f_2(X)$ in equation (4.4.26) was optimised by BFAVP and the minimal power loss F_{L} obtained is given in Table 4.8. The optimal location of the FACTS devices and their corresponding parameters, found by BFAVP, are listed in Table 4.9.

	BFAVP	PSO	GA
Power loss (MW)	**13.8533**	15.1026	18.8402

Table 4.8: The least power loss obtained by BFAVP, PSO, and GA for the IEEE 14-bus test case

In order to compare BFAVP with other optimisation algorithms, PSO and GA were applied to minimise the real power loss of the network by optimally allocating the multi-type FACTS devices as well. The least power loss reached by the three algorithms of 20 runs was given in Table 4.8. According to Table 4.8, it can be concluded that BFAVP achieves the best result in the case of installing multi-type FACTS devices in the IEEE 14-bus test case. Compared with the original IEEE 14-bus test case without FACTS devices installed, the total real power loss has been decreased from 19.4121 MW to 13.8533 MW, which is a reduction of about 30%.

FACTS devices	Location (bus)		Device setting		Cost C_F ($)
	From	To			
SVC	7	9	Q_{SVC}:	35.613	
TCSC	1	5	X_{TCSC} :	-0.0620	
	2	4		-0.0035	
	4	7		-0.0153	5.1750×10^6
	6	11		0.0015	
	6	12		-0.0663	
	13	14		-0.1440	
TCPST	2	3	θ_{TCPST}:	0.1417	

Table 4.9: Optimal location and setting of FACTS devices to minimise the real power loss in the IEEE-14 bus test case

Multi-objective case studies

Furthermore, the performance of BFAVP has been verified on the IEEE 30-bus test case. The system consists of 48 branches, 6 generator buses and 22 load-buses. The single-line diagram of the IEEE 30-bus test case is shown in Figure 4.2 and the system parameters are given in [4]. Simulations were carried out for the optimal location of single or multi-type FACTS devices in the system, which was handled as a multi-objective optimisation problem using BFAVP with the dominance method adopted. Research work in [6] and [142] compared the FACTS devices allocation results obtained by multi-objective GA, PSO, GSO and BFAVP. According to the comparison, there is not remarkable differences among the Pareto-fronts obtained by these algorithms. As a result, this experiment only demonstrates the improvement of the power system by allocating FACT devices using BFAVP.

When minimising the operation cost and power loss, multi-type FACTS devices were placed in the IEEE 30-bus test case [4]. The SLP λ was set at 1.2, which reflects a 20% increment in power demand. BFAVP was applied to find optimal allocation of multi-type FACTS devices, and a set of non-

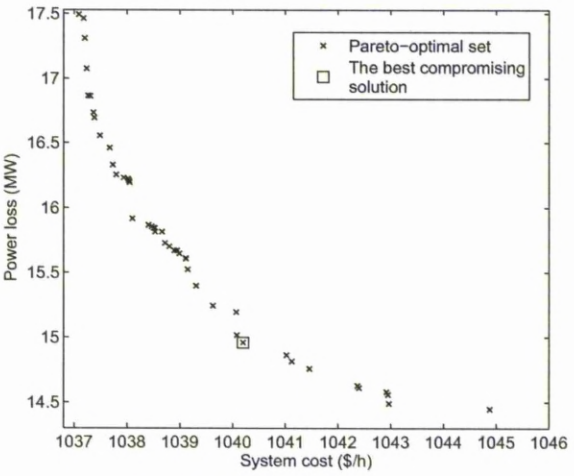


Figure 4.13: The Pareto-front of BFAVP in a single run for optimal placement of FACTS devices

dominated solutions were found. These non-dominated solutions constitute the Pareto-optimal solution set. The diversity of the Pareto-optimal solution set over the trade-off surface is shown in Figure 4.13.

The operation cost in Figure 4.13 reflects the evaluation value of $f_1(X)$ in equation (4.4.25), which is the sum of total fuel cost and installation cost of FACTS devices. The power loss in Figure 4.13 is calculated by equation (4.4.19) considering the system constraint violation penalty. From the range of the values of power loss, it can be seen that the Pareto-optimal solution set plotted in Figure 4.13 contains the solutions found within the system security margin.

Each member of the Pareto-optimal solution set was evaluated by the membership functions given in [132]. The one having the maximal value for membership function can be extracted as the best compromising solution, which is marked by a box (□) in Figure 4.13. The location of FACTS devices concerning the best compromising solution, with their settings of control variables are given in Table 4.10.

FACTS devices	Location (bus)		Device setting		Costs ($/h) & Power loss (MW)	
	From	To				
SVC	8	28	Q_{SVC}:	51.1987		
TCSC	2	6	X_{TCSC} :	0.0023	C_{FC}:	1033.0323
	6	10		-0.1494		
	12	13		-0.0117		
	12	14		-0.0617		
	12	15		-0.0364	C_{FH}:	7.1660
	14	15		-0.0732		
	18	19		-0.0339		
	10	22		-0.0352		
	25	26		-0.0891		
	28	27		-0.2529	P_L:	14.9583
TCPST	6	8	θ_{TCPST}:	-0.0265		

Table 4.10: The best compromising solution and its optimal location and setting of FACTS devices in the IEEE 30-bus test case

BFAVP was also applied to find the optimal location and setting for one single type FACTS device in the system and for the minimisation of the overall cost and power loss within the system security margin. For each case of single type FACTS devices, a Pareto-optimal solution set was obtained by BFAVP, and the best compromising solution was found among each group of solutions in the optimal set. For comparison purpose, the results of single and multi-type FACTS devices, including system fuel cost C_{FC}, installation cost of FACTS devices C_{FH} and real power loss P_L, are listed in Table 4.11. As indicated in Table 4.11, the TCSC has greater impact on minimising power loss compared with other FACTS devices, while the TCPST contributes more to decreasing system total fuel cost. A compromise between system overall cost and power loss is obtained by allocation of multi-type FACTS devices, which is achieved by applying BFAVP to solve the multi-objective optimisation problem.

FACTS devices	C_{FC} \$/h	C_{FH} \$/h	P_L MW
SVC	1026.1520	6.3307	16.5026
TCSC	1042.8241	0.6746	13.9936
TCPST	1021.0719	0.2624	18.1320
UPFC	1031.0045	1.4345	15.4172
Multi-type	1033.0323	7.1660	14.9583

Table 4.11: The results of best compromising solutions for single and multi-type FACTS devices

When minimising operation cost and system loadability, suppose that the total number of FACTS devices to be installed is given previously. The objective was to find the maximal load parameter of the IEEE 30-bus test case with the help of FACTS devices. At the same time, these FACTS devices were optimally placed in the system to minimise the overall operation cost. This case was formulated as a multi-objective optimisation problem which is dealt with by BFAVP to find the optimal allocation of FACTS devices. In this case, the value of the SLP, λ, was added as a control variable and bounded in $[1, 2]$; and in total eight FACTS devices were placed in the system to increase the SLP.

The multi-objective optimisation problem was solved by BFAVP placing multi-type FACTS devices in the system. The Pareto-optimal solution set within the system security margin is obtained and given in Figure 4.14. The solution shown at the bottom right of Figure 4.14, which is marked by ∇, has the maximal SLP: $\lambda_{max} = 1.4231$. Among the Pareto-optimal solution set, the best compromising solution between the system overall cost and SLP is found and marked by $\square$. The detailed location and setting of the eight FACTS devices installed are given in Table 4.12. They consist of one SVC, four TCSCs and three TCPSTs. Each of them is placed at a different branch in the system and the SLP for the best compromising solution, λ_c, is enhanced to 1.2821.

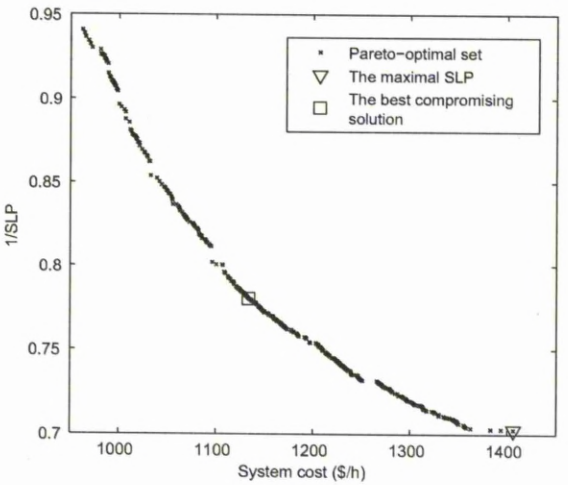


Figure 4.14: The Pareto-front set of BFAVP in a single run for allocation of multi-type FACTS devices

FACTS devices	Location (bus)		Device setting		Costs ($/h) & Power loss (MW)	
	From	To				
SVC	6	7	Q_{SVC}:	37.6088		
TCSC	5	7	X_{TCSC} :	-0.0213	C_{FC}:	1131.6469
	12	16		-0.0573		
	15	23		-0.0886	C_{FH}:	2.0153
	19	20		-0.0270		
TCPST	2	6	θ_{TCPST}:	-1.6903		
	6	8		-2.3312	λ:	1.2821
	6	10		-1.5553		

Table 4.12: The best compromising solution found by BFAVP for optimal allocation of multi-type FACTS devices

For comparison purposes, the same problem was also solved by BFAVP with one single type FACTS device installed in the system each time. A Pareto-optimal solution set was obtained for each type of FACTS device. Among each of the Pareto-optimal solutions, the best compromising solution and the

solution having the maximal SLP were found and the results including the fuel cost, the installation cost of FACTS devices and the SLP, are given in Table 4.13. According to Table 4.13, it can be seen that the placement of the UPFC can achieve the largest SLP but its installation cost is the highest. SVC achieves the second highest SLP followed by the UPFC. The TCSC and TCPST give a lower SLP and their costs are cheaper. Compared with the installation of a single type FACTS device, the installation of multi-type FACTS devices is a wiser choice. Under the latter condition, a larger SLP can be obtained with a more reasonable cost. Furthermore, among the best compromising solutions for all cases, the case of placing multi-type FACTS devices provides the highest SLP with the least operation cost.

FACTS devices	Case	λ \$/h	C_{FC} \$/h	C_{FH}
SVC	λ_{max}	1.4217	1327.4926	145.9535
	λ_c	1.2684	1127.1230	19.9627
TCSC	λ_{max}	1.3416	1197.9395	27.3683
	λ_c	1.2012	1030.2312	0.5489
TCPST	λ_{max}	1.3966	1288.6468	2.0993
	λ_c	1.2044	1025.2327	2.0995
UPFC	λ_{max}	1.4489	1363.9886	214.1062
	λ_c	1.2811	1131.5116	85.0187
Multi-type	λ_{max}	1.4231	1328.4659	77.9569
	λ_c	1.2821	1131.6469	2.0153

Table 4.13: The results of the best compromising solutions for single and multi-type FACTS devices

4.4.5 Summary

This section presented a novel method for the optimal location of multi-type FACTS devices in a power system using BFAVP. Four types of FACTS devices

were chosen and modelled. The optimisation of the locations of the devices and their setting parameters were undertaken using BFAVP and this has been compared with GA and PSO. It can be seen from the experimental results that with BFAVP, that it is possible for utility companies to place multi-type FACTS devices in a transmission system such that the optimal reactive power planning can be achieved and the system real power loss can be minimised. It has verified that BFAVP is suitable for finding the optimal placement of multi-type FACTS devices in large-scale power systems, and the result of the reactive power planning for the power systems can be improved using this novel biologically inspired algorithm.

4.5 Dynamic Harmonic Estimation

4.5.1 Background

BFAVP was used to estimate the harmonic parameters in a dynamic environment. Power electronic devices feed harmonic currents into the power system, which known as harmonic pollution. Harmonic pollution has serious hazards on electrical equipments, such as increasing the loss of transmission cables from harmonic current, reducing transmission capacity, increasing the motor loss and heat, limiting overload capacity and efficiency and lifetime, and even damaging the equipments [111]. In electric power networks, the increasing use of power-electronic equipment has caused much more harmonic pollution [143], which significantly deteriorates the power quality. In order to reduce the harmonic pollution, it is necessary to estimate the parameters of the harmonics. With the estimated parameters, such as amplitudes and phases, the harmonic components can be compensated for by injecting the corresponding portion into a power system.

There are three types of harmonics. Integral-harmonics are spectral components whose frequencies are integer multiples of the system's fundamental frequency. Inter-harmonics are components at frequencies that are not integer multiples of the system's fundamental frequency. Sub-harmonics are inter-

harmonics with frequencies lower than the fundamental frequency. In the past few decades, various approaches have been proposed to estimate the parameters of these harmonics.

For integral-harmonics, the most widely used approach is the fast executable algorithm derived form the discrete Fourier transform (DFT). However, the DFT-based algorithms do not perform stably if certain undesirable conditions are given. A successful algorithm is the Kalman filtering approach, which is simple, linear and robust [144]. However, it requires *a prior* knowledge of the statistics of the signal and the state matrix needs to be defined accurately as well. As for inter-harmonics and sub-harmonics, which also appear frequently in electrical signals and severely affect the accuracy of DFT-based algorithms, few attempts have been made to estimate them, except for the work of [145], which proposes an optimisation algorithm for the estimation of not only the phases and amplitudes of integral-harmonics but also the phase, amplitude and frequency of the inter-harmonics.

However, despite all of the advantages and disadvantages of the algorithms discussed above, they are basically static estimation methods. In most cases, in addition to the harmonics, the fundamental frequency also deviates. Moreover, its deviation rate varies with time. As a result, it is necessary to estimate the parameters of the harmonics as well as the deviation of the fundamental frequency. Furthermore, for practical applications, the estimation needs to be in real-time. Inspired by the work of [145], this section proposes an optimisation algorithm to solve the harmonic estimation problem.

The process of harmonic parameter estimation is described in Section 4.5.2. In order to work in a dynamic environment, BFAVP introduces an environment adaptation, which is modelled by the growth during the lag phase, to ensure the algorithm is adaptive in dynamic environments. The improved algorithm is applied to estimate the parameters of the harmonics in dynamic environments in Section 4.5.3.

4.5.2 Harmonic estimation in dynamic environments

This section presents the estimation process of the parameters of harmonics in dynamic environments using the non-linear optimisation algorithm and the linear least square method. An electrical signal $Z(t)$ can be described as:

$$Z(t) = \sum_{n=1}^{N} A_n \sin(w_n t + \phi_n) + v(t), \tag{4.5.1}$$

where $n = 1, 2, \ldots N$ represents the order of the harmonic, A_n, ϕ_n and w_n are the amplitude, phase angle and angular frequency of the n^{th} harmonic, respectively, $w_n = 2\pi f_n$, and $v(t)$ is the additive noise. It should be noted that in a dynamic environment, w_n varies with t.

To estimate the ϕ_n, w_n and A_n of each harmonic, the following function is constructed:

$$\hat{Z}(t) = \sum_{n=1}^{N} \hat{A}_n \sin(\hat{w}_n t + \hat{\phi}_n), \tag{4.5.2}$$

where $\hat{A}_n$, $\hat{w}_n$ and $\hat{\phi}_n$ are the estimation of A_n, w_n and ϕ_n, respectively. Thereby, the original signal can be represented as:

$$Z(t) = \hat{Z}(t) + r(t) = \sum_{n=1}^{N} \hat{A}_n \sin(\hat{w}_n t + \hat{\phi}_n) + r(t), \tag{4.5.3}$$

where $r(t)$ is the estimation residual that indicates the difference between $Z(t)$ and $\hat{Z}(t)$. The estimation goal is to search for the best $\hat{\phi}_n$, $\hat{w}_n$ and $\hat{A}_n$ such that $r(t)$ is the smallest with the condition of N being determined.

Because the phases of the harmonics in the model are non-linear and are restricted within $[0, 2\pi)$, the recursive optimisation algorithm is utilised to estimate the values of ϕ_n as well as w_n. Once the phases and the frequencies are estimated in each iteration, the amplitude A_n is obtained by the standard least square method.

The signal $Z(t)$ is converted to its discrete version, $Z(m)$, by sampling. Instead of considering that the frequencies w_n that vary sample by sample, it is assumed that they vary cycle by cycle. Therefore, the optimisation process uses the data of one cycle in each iteration. Supposing a cycle of $Z(m)$ consists

of M samples, the discrete linear model is given as:

$$\begin{aligned} Z_c(m) &= H_c(m) \cdot A + v(m), \; m = 1, 2, \ldots, M, \\ H_c(m) &= \begin{bmatrix} \sin(w_{c1}t_{c1} + \phi_1) & \ldots & \sin(w_{cN}t_{c1} + \phi_N) \\ \sin(w_{c1}t_{c2} + \phi_1) & \ldots & \sin(w_{cN}t_{c2} + \phi_N) \\ \vdots & \ldots & \vdots \\ \sin(w_{c1}t_{cm} + \phi_1) & \ldots & \sin(w_{cN}t_{cm} + \phi_N) \\ \vdots & \ldots & \vdots \\ \sin(w_{c1}t_{cM} + \phi_1) & \ldots & \sin(w_{cN}t_{cM} + \phi_N) \end{bmatrix}, \end{aligned} \tag{4.5.4}$$

where $w_{cn} = 2\pi n f_c$ with f_c the fundamental frequency of the c^{th} cycle, $Z_c(m)$ as the m^{th} sample of the c^{th} cycle with additive noise $v(m)$, $A = [A_1 \; A_2 \; \cdots \; A_N]^T$ is the vector of the amplitudes needing to be estimated, $H_c(m)$ is the system structure matrix, w_{cn} is the frequency of the n^{th} harmonic of the c^{th} cycle, and t_{cm} is the $[(c-1)M + m]^{\text{th}}$ sampling time. As stated above, the task is to find the best $\hat{\phi}_n$, $\hat{w}_{cn}$, and $\hat{A}_n$ to minimise the difference between $Z_c(m)$ and $\hat{Z}_c(m) = \hat{H}_c(m)\hat{A}$.

After the values of $\hat{\phi}_n$ and $\hat{w}_{cn}$ are evaluated using a certain optimisation algorithm, $\hat{H}_c(m)$ is calculated. Assuming that $\hat{H}_c(m)$ is a full-rank matrix, the estimation of $\hat{A}$ can be obtained via the standard least square algorithm as:

$$\hat{A} = [\hat{H}_c^T(m) \cdot \hat{H}_c(m)]^{-1}\hat{H}_c^T(m)Z_c(m), \tag{4.5.5}$$

which ensures that the estimation of the signal:

$$\hat{Z}_c(m) = \hat{H}_c(m)\hat{A} \tag{4.5.6}$$

is the best given the conditions of $\hat{\phi}_n$ and $\hat{w}_{cn}$. However, the values of $\hat{\phi}_n$ and $\hat{w}_{cn}$ are not the best solution and need to be optimised. Therefore, in the next iteration, $\hat{\phi}_n$ and $\hat{w}_{(c+1)n}$ are updated using a certain optimisation algorithm according to the cost function F, which is defined as:

$$F = \sum_{m=1}^{M}(Z_c(m) - \hat{Z}_c(m))^2. \tag{4.5.7}$$

The process is repeated until the final convergence condition is reached.

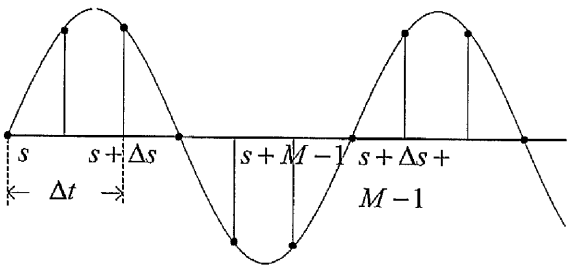


Figure 4.15: The sample selecting strategy

Suppose it uses Δt ms to perform the first iteration. Suppose further that Δt ms corresponds to Δs samples of the signal $Z(m)$. In the next iteration, a set of data from $Z(s+\Delta s)$ to $Z(s+\Delta s+M-1)$ is used to update $\hat{\phi}_n$ and $\hat{f}_{s+\Delta s}$. Figure 4.15 briefly illustrates this process. Theoretically, the iteration repeats until the final convergence condition is reached, which is either $J \leq \varepsilon$ with ε as a predefined constant or the maximum iteration K is reached. In practice, it ends when all the samples are manipulated.

In summary, the decision variables of the harmonic parameter estimation in this experiment are the amplitudes, phase angles and angular frequencies of harmonics. The objective of the optimisation is to minimise the difference between the reconstructed signal and the measured data.

4.5.3 Experimental studies and results

In a dynamic environment, when the environment changes, the nutrients are redistributed and the bacteria need to adapt. Compared to the environment of the previous iteration, in which a large amoount of nutrients have been absorbed by bacteria, the new environment has more nutrients; thus, it allows bacterial growth behaviour. There are four phases in holistic growth. The first one is the lag phase, a period of slow growth when bacteria acclimatise to the nourishment environment. The lag phase has high biosynthesis rates because

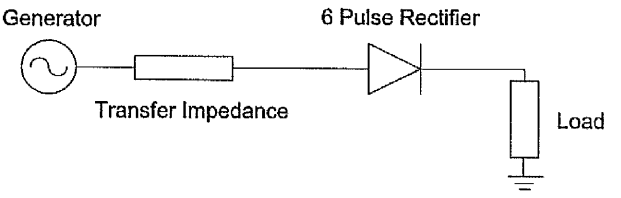


Figure 4.16: A simple power system: a two-bus architecture with a six-pulse full-wave bridge rectifier supplying the load

enzymes and nutrient transporters are produced [73].

In BFAVP, the reaction to the change in the environment results in a short lag phase period. During the lag phase, in order to prevent BFAVP from converging to the previous optima, which was obtained before the changes in the environment, the global best solution is updated, which can be modelled by:

$$X_{\text{best}} = X_l^k|_{F(X_l^k)=\min_p F(X_p^k)}, \tag{4.5.8}$$

where l is the index for the bacterium having the global best solution, and $F(X_l^k)$ is its fitness value. The new global best solution is used for quorum sensing of the $(k+1)^{\text{th}}$ iteration.

In this section, BFAVP is applied to estimate the harmonic parameters in a a dynamic environment, although it was not designed exclusively for this type of problem. For the purpose of comparison, the same experimental module as used in [146] was employed to produce test signals. Figure 4.16 describes a simple power system, which comprises a two-bus three-phase system with a full-wave six-pulse bridge rectifier at the load bus. The test signal, denoted $Z_o(t)$, is a distorted voltage waveform sampled from the terminal of the load bus in the system. The frequencies of the harmonics of the test signal are listed in Table 4.14. The signal was sampled at the rate of 64 points per cycle.

Harmonic Order	Amplitude (p.u.)	Phase (°)
Fundamental $n = 1$ (50 Hz)	1	0
$n = 2$ (100 Hz)	0.07	-2.02
$n = 3$ (150 Hz)	0.05	82.1
$n = 4$ (200 Hz)	0	\
$n = 5$ (250 Hz)	0.04	7.9
$n = 6$ (300 Hz)	0	\
$n = 7$ (350 Hz)	0.03	-147.1
$n = 8$ (400 Hz)	0.01	162.6

Table 4.14: Harmonic content of the test signal $Z_o(t)$

To evaluate the performance of BFAVP, a DFT-based scheme and GA were also employed in the experimental studies to compare their estimation results. The setting of the parameters of GA follows the design of [145]. For the DFT-based scheme, the approximation of the fundamental frequency was needed as *prior* information. It first creates a comb filter to remove harmonics other than the approximate fundamental harmonic so as to find a new fundamental frequency. Then, the comb filter is redesigned, and the approximate fundamental harmonic is updated. These steps are repeated until the approximate fundamental frequency converges. After the fundamental harmonic is determined, it is easy to obtain the approximation of the frequencies of other harmonics because they are the multiples of the fundamental frequency. Therefore, each harmonic can be calculated separately following a similar recursive procedure.

Scenario I

In this case, the deviated fundamental frequency f of the test signal is assumed to vary in the range of 49 Hz to 51 Hz, and follows the equation:

$$f(t) = 50 + y(t), \tag{4.5.9}$$

where $y(t)$ is a square waveform:

$$y(t) = \begin{cases} 1 & 0 \leq t < r/2 \\ -1 & r/2 \leq t < r \end{cases}, \tag{4.5.10}$$

and

$$y(t + r) = y(t). \tag{4.5.11}$$

Here r is a coefficient for indicating the period of the fundamental frequency deviation. The sampling rate is 64 samples per cycle, *i.e.*, $M = 64$. Hence, each fitness value is calculated based on 64 samples. In this experiment, two cases have been studied, and the period of the fundamental frequency deviation is $r = 6.4$ s and $r = 3.2$ s, respectively.

Figures 4.17 and 4.18 illustrate the trace of the frequency deviation by BFAVP under different conditions of the coefficient r. Obviously, BFAVP estimates the deviation of the fundamental frequency successfully. It should be noted that all the results presented in this section are the average of 30 runs. The test signal lasts 20 s and about 200 iterations are performed during this interval, which means it takes roughly 100 ms to estimate f. Since the power system signal has a period of 20 ms and f will not change too fast, it is possible to design a practical system to perform online identification using hardware such as DSP.

Figure 4.19 shows the original signal and the signals reconstructed by using the parameters identified by BFAVP, GA, and DFT. To view the difference more clearly, a segment of Figure 4.19 is enlarged in Figure 4.20. As can be observed from these figures, the signal reconstructed by using the parameters estimated by BFAVP matches the original signal, while the one using the parameters identified by GA is slightly different compared to the original one. On the contrary, there is a considerable difference between the reconstruction result of DFT and the original signal. Obviously, DFT fails to trace the change in the fundamental frequency due to the lack of a dynamic mechanism.

In order to analyse the reconstruction results quantitatively, the mean errors of the signals reconstructed using the parameters identified by BFAVP, GA,

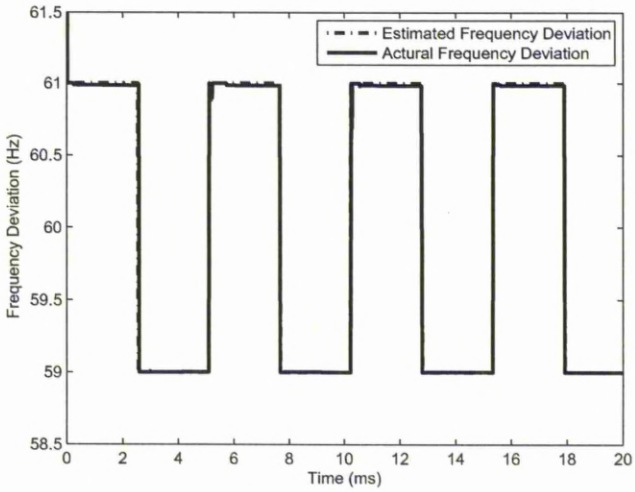


Figure 4.17: The trace of frequency deviation by BFAVP ($r = 6.4$ ms)

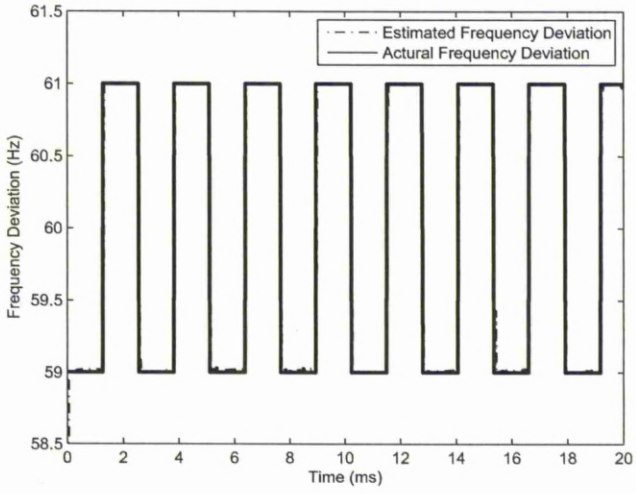


Figure 4.18: The trace of frequency deviation by BFAVP ($r = 3.2$ ms)

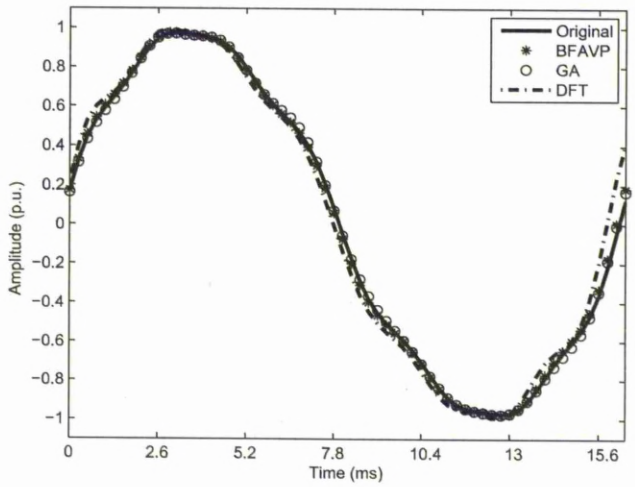


Figure 4.19: The comparison of the original signal ($r = 1$) and the signals reconstructed using the parameters estimated by GA, DFT, and BFAVP

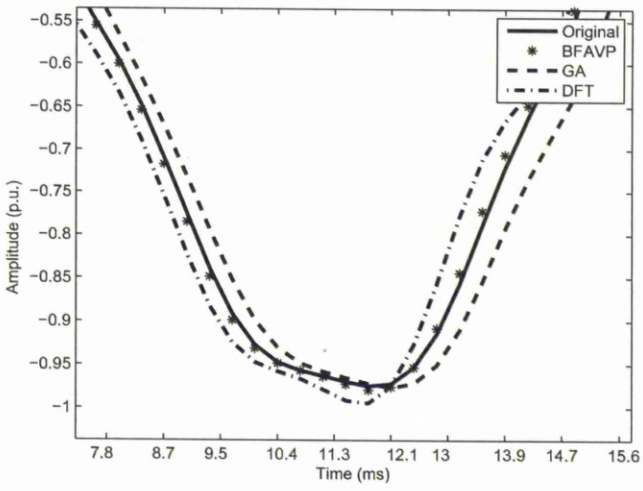


Figure 4.20: An enlarged segment of Figure 4.19

and DFT are listed in Table 4.15. The mean error is the average error of each sample, which can be expressed as:

$$e = \frac{J}{M} = \frac{\sum_{m=1}^{M} (Z_c(m) - \hat{Z}_c(m))}{M}. \tag{4.5.12}$$

The comparison between BFAVP and GA is based on the same numbers of evaluations. From the table, it can be concluded that by using the harmonic parameters obtained by BFAVP, the reconstructed signal has a smaller error than the other two algorithms. DFT fails to obtain the correct parameters to reconstruct the signal in the dynamic environment. When the environment changes, the individuals consult the old schema to generate a new position. Therefore, due to the obstruction of old fitness values, the individuals may be trapped at an incorrect position.

r	BFAVP	GA	DFT
6.4	$\mathbf{2.3 \times 10^{-4}}$	0.0726	0.3712
3.2	$\mathbf{4.1 \times 10^{-4}}$	0.0909	0.4608

Table 4.15: Comparison of the mean error (e) among BFAVP, GA, and DFT in Case I

Scenario II

In this case, a sawtooth waveform is added to the deviated fundamental frequency, f_0. The range of f_0 is [59,61] Hz and its mathematical expression is given as follows.

$$f_0(t) = 2(t/s - \text{floor}(t/s) + 0.5), \tag{4.5.13}$$

where s is a coefficient that indicates the period of the sawtooth waveform, and the operator floor(X) rounds the elements of X to the nearest integers towards negative infinity.

Figures 4.21 and 4.22 illustrate the trace of frequency deviation by BFAVP with sawtooth waveform interference. In the figures, it can be seen that BFAVP

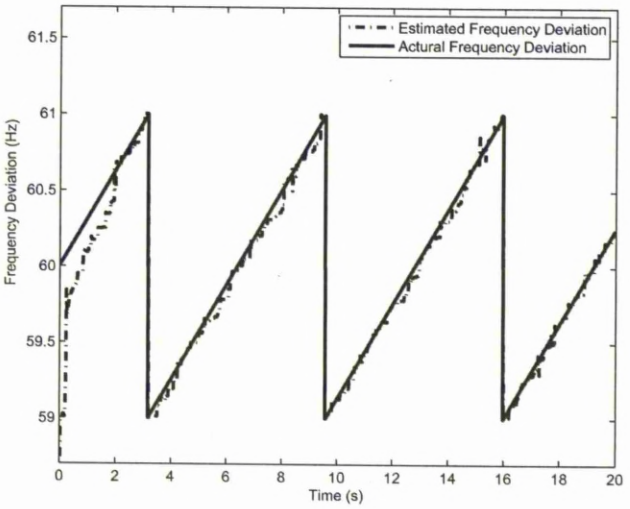


Figure 4.21: The trace of frequency deviation by BFAVP ($s = 6.4$)

traces the change of the environment rapidly. It takes 1.96 ms ($s = 6.4$) and 1.38 ms ($s = 3.2$) for BFAVP to converge, respectively. After the convergence period, BFAVP traces the change of the environment iteration by iteration.

It is hard for most of the optimisation algorithms to obtain an accurate estimation of the deviated frequency when the fitness values of the entire population are increasing because the searching history interferes with the performance of algorithms. In contrast, a decreasing fitness value in the searching history gives positive assistance. However, as delineated in Figures 4.21 and 4.22, BFAVP is adaptive to the changing environment despite the kind of searching history it has.

Figure 4.23 illustrates the comparison of the original signal and the signals reconstructed using the parameters estimated by BFAVP, GA and DFT, respectively. A segment of Figure 4.23 is enlarged and is shown in Figure 4.24. The mean errors of these reconstructed signals are listed in Table 4.16. The same conclusion can be drawn from the figures and the table as in the previous simulation study.

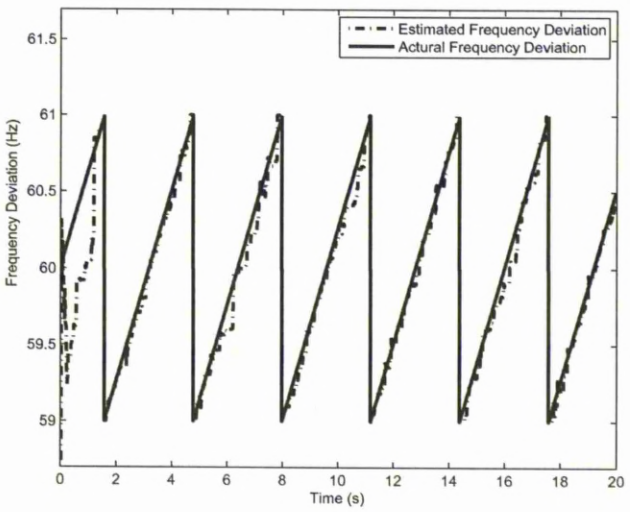


Figure 4.22: The trace of frequency deviation by BFAVP ($s = 3.2$)

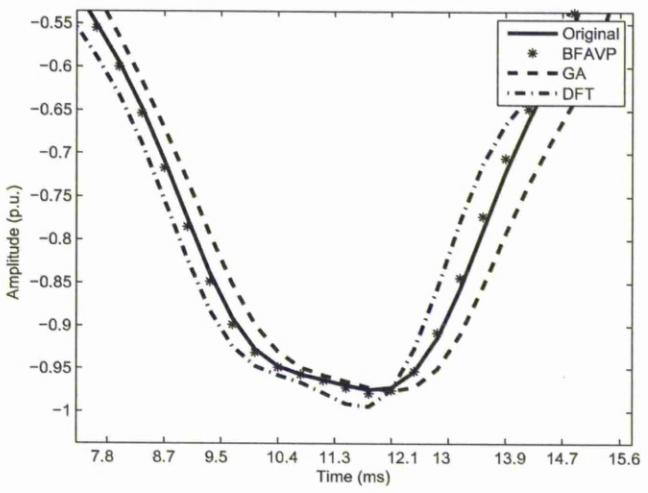


Figure 4.23: The comparison of the original signal ($s = 6.4$) and the signals reconstructed using the parameters estimated by GA, DFT, and BFAVP

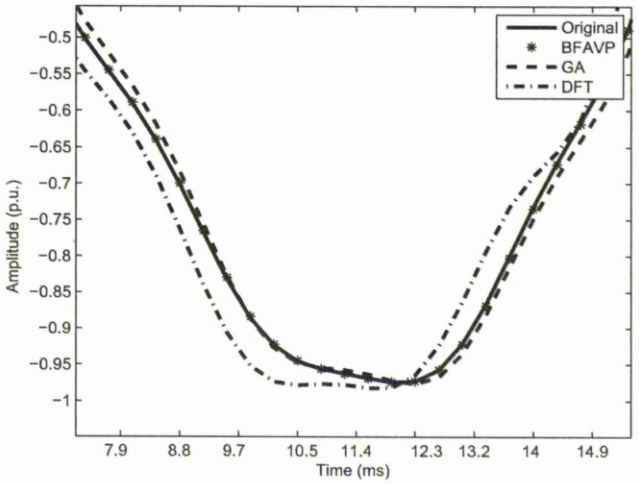


Figure 4.24: An enlarged segment of Figure 4.23

s	BFAVP	GA	DFT
6.4	$\mathbf{8.9375 \times 10^{-4}}$	0.0455	0.3377
3.2	**0.0015**	0.0521	0.3460

Table 4.16: Comparison of the mean error (e) among BFAVP, GA, and DFT in Case II

4.5.4 Summary

In this study, BFAVP was applied to estimate the frequencies, amplitudes and phases of the harmonics as well as the deviation of the fundamental frequency, which places the problem in a dynamic environment, of a voltage or current waveform for power quality monitoring. Two different dynamic environments were considered in the experimental studies, and the proposed algorithm achieved an improved performance over GA and DFT-based schemes in both cases. The comparison of the numerical estimation results among BFAVP, GA and DFT were also presented to demonstrate the advantage of BFAVP in terms of accuracy. Experimental results have shown that BFAVP is suitable

for the optimisation of the parameters of the harmonics in a dynamic environments. Moreover, the convergence rate of BFAVP is high enough for the algorithm to be developed for on-line estimation.

4.6 Conclusion

In this Chapter, BFAVP was applied to four real-world power system optimisations. In contrast to the conventional EAs, BFAVP has a varying population which was introduced by simulating the phenomenon of cells' division and this contributes significantly to global search. The applications of the OPF problem, allocation of FACTS devices and harmonic parameter estimation have been considered complicated optimisation problems in power systems. This chapter demonstrated the results of solving these problems using BFAVP, which was proposed in Chapter 2. The results not only have shown its advantages in tracking power system optimisation problems in both static and dynamic environments but also have verified its potential in other real-world applications.

Chapter 5

Applications of PBO to Power System

This chapter applies Paired-bacteria Optimiser (PBO) to solve two real-world power system optimisation problems. The first application aims to optimise the fuel cost in a dynamic environments. Then a stochastic Optimal Power Flow (OPF) is adopted to describe the real-world power system environment, and PBO is applied to optimise the distribution of the stochastic fuel cost.

5.1 Introduction

Most research assumes the power system is static during the optimisation process [119][121][88]. However, the properties such as the loads, generation capacities and network connections in a power system are always changing with time. In this chapter, PBO proposed in Chapter 3 is introduced to solve the OPF problem with dynamic loads. To meet the demand of online optimisation, PBO is improved to adapt to the environmental changing. The experimental results have demonstrated that PBO is able to track the environmental change rapidly and has a superior performance over Particle Swarm Optimiser (PSO).

Furthermore, a power system model with stochastic real power loads is

proposed in this chapter. The model is able to predict a fuel cost distribution based on load rated values. Evolutionary algorithms (EAs) are notorious for the intensive computation caused by a large amount of evaluations of the objective function required by all individuals. To reduce the computational time on stochastic OPF, PBO is adopted to solve the expensive optimisation problem in this chapter. The PBO is applied to a stochastic optimal voltage control and fuel cost reduction problem, which is a crucial part of power system planning. The experimental results have shown that PBO has a more reliable performance than the widely applied EAs.

The rest of this chapter is organised as follows. In Section 5.2 PBO proposed in Chapter 3 is applied to minimise the fuel cost in a dynamic OPF. Then the algorithm is applied to optimise the fuel cost of a stochastic OPF in Section 5.3. Conclusions about the applications of PBO are drawn in Section 5.4.

5.2 Optimal Dynamic Power Flow

5.2.1 Background

The OPF has been conventionally regarded as a static optimisation problem, based on many assumptions made about power system operation conditions and load patterns. However, in practice, power system operations are always subject to relevant uncertainty factors because the loads, generation capacities and network connections in a power system are always changing over time. It is understood that the online OPF computation takes place every 10$\sim$20 minutes for power system economic dispatch and secondary control purposes. It is also noted that after the completion of this online task, the state of a power system has already changed, since over such a long period of time, the system loads vary. However, in past few decades, an assumption has been made in that no change is applied to power system states during the duration of the online OPF computation, and the results obtained from this online computation task are accurate and can be used for many other tasks of power system operation such as network and generation control, system dis-

patch, *etc.* The assessment regarding the errors caused by this assumption has never been computed.

In order to waive the above assumption, developing a methodology that could be used to trace the optimum solution of power flow in a dynamic environment is desired. In the first step, real power demands with various load changes are considered in the experimental test case. Then PBO assumes the load is a constant in each iteration as a static optimisation problem. Global optima are also updated during each change to prevent individuals from being attracted to previous optima before the environmental change occurs.

Section 5.2.2 gives a description of the OPF formulations in dynamic environments. In Section 5.2.3, PBO has been evaluated on the IEEE 30-bus test case. The experimental results and analyses are also included.

5.2.2 Optimal power flow in dynamic environments

The OPF problem in a dynamic environment is expressed as:

$$\min \quad F(u_{\mathrm{d}}, u_{\mathrm{c}}, d(t)) \tag{5.2.1}$$

$$\text{s.t.} \quad G(u_{\mathrm{d}}, u_{\mathrm{c}}, d(t)) \leq 0 \tag{5.2.2}$$

$$H(u_{\mathrm{d}}, u_{\mathrm{c}}, d(t)) = 0 \tag{5.2.3}$$

Different from equation (4.2.1), (4.2.2), and (4.2.3), the formulation in a dynamic environment introduces a $d(t)$ to describe the dynamic loads in time t, which is expressed as:

$$d(t) = (d_1(t), d_2(t), ..., d_{N_D}(t)), \tag{5.2.4}$$

where $d_i(t)$ indicates the dynamic load of the i^{th} bus at time t. The real power demand P_{D} in equation (4.2.6) in the dynamic environment is calculated as:

$$P_D(t) = P_{\mathrm{D}_0} + d(t), \tag{5.2.5}$$

where P_{D_0} indicates the real power demand of the original rated value. As P_{D} is changing with time t, one set of control variables and one corresponding fuel cost can be estimated by PBO in each iteration.

5.2.3 Experimental studies and results

The PBO algorithm was tested on the standard IEEE 30-bus test system, which was adopted from [4]. The layout of the test case is illustrated in Figure 4.2. To operate such a complex case, 23 variables were chosen to be the control variables, which are defined in Section 4.2.2. In order to simulate the dynamic power flow variation, the real power demand at branch 7 was varied in probability during the simulation process. However, when solving the power flow equation inside an iteration, the value is fixed to a constant. In this case, the objective function is the total fuel cost, which is formulated in Section 4.2.3

To evaluate the performance of PBO in dynamic OPF problems, various environmental changes were applied in the experimental studies, which could be divided into three ranges:

- Case I – Slow level of environmental changes
- Case II – Intermediate level of environmental changes
- Case III – High level of environmental changes

The level of changes is reflected in the frequency of changes in the environment, which is defined as a probability τ. The environmental change rates in these three cases were 0.005, 0.01, and 0.05. In the simulation, τ indicated the probability of occurrence of environmental changes after each iteration. The environmental changes were simulated with P_{D_7} varying by 20% of the initial real power demand of branch 7.

The performance of PBO was compared with that of Particle Swarm Optimiser (PSO) [35] in this section. For PSO, a population of 50 individuals was used. The inertia weight, ω, was set to 0.73, and the acceleration factors c1 and c2 were both set to 2.05 which follows the recommendations from [36]. For PBO, the population size was always 2 individuals in each generation. The inertia weight, γ, was set to 0.85. The two-value random factor, c_c, was set to 0.02 and 20. The quorum sensing attraction factor, c_q, was set to 2.05.

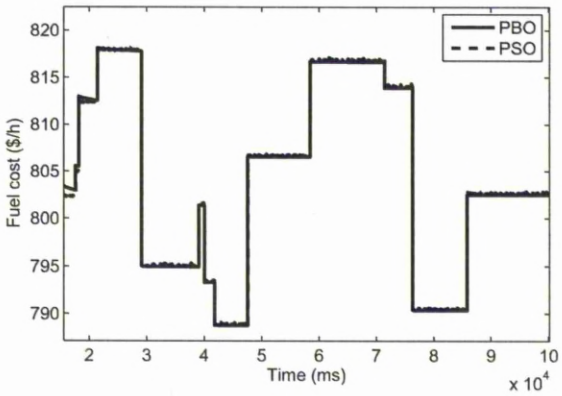


Figure 5.1: The fuel cost of the system operated by the parameters from PBO and PSO ($\tau = 0.005$)

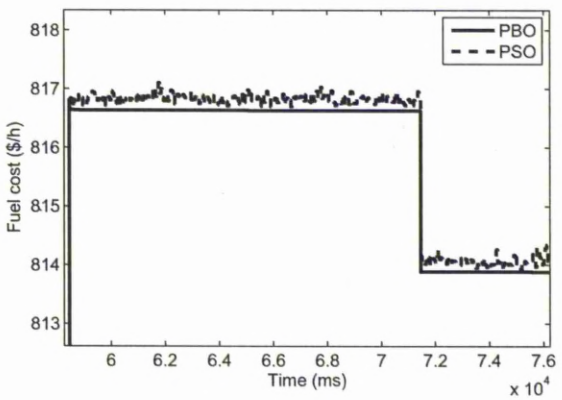


Figure 5.2: The detailed fuel cost of the system operated by the parameters from PBO and PSO ($\tau = 0.005$)

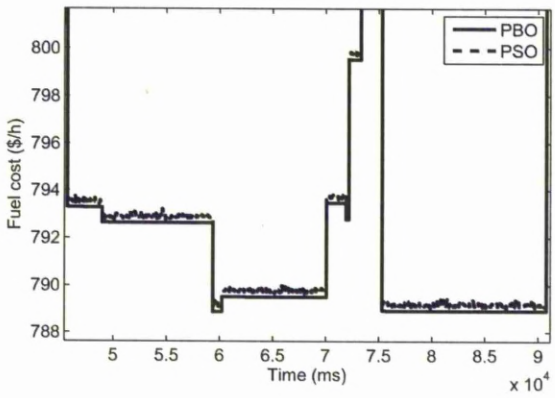


Figure 5.3: The fuel cost of the system operated by the parameters from PBO and PSO ($\tau = 0.01$)

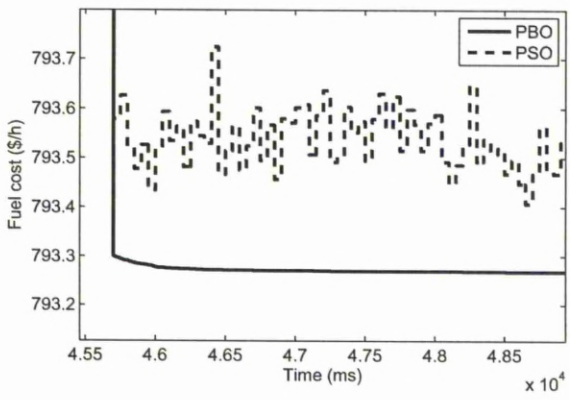


Figure 5.4: The detailed fuel cost of the system operated by the parameters from PBO and PSO ($\tau = 0.01$)

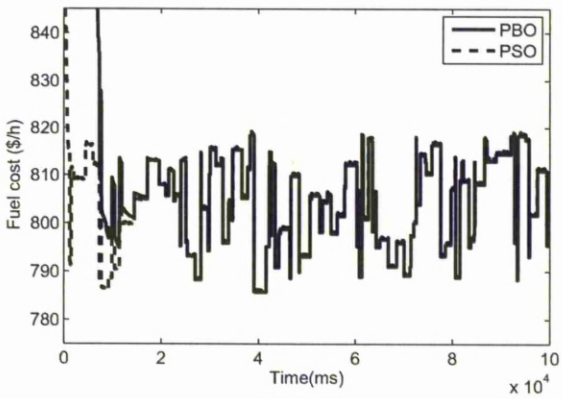


Figure 5.5: The fuel cost of the system operated by the parameters from PBO and PSO ($\tau = 0.05$)

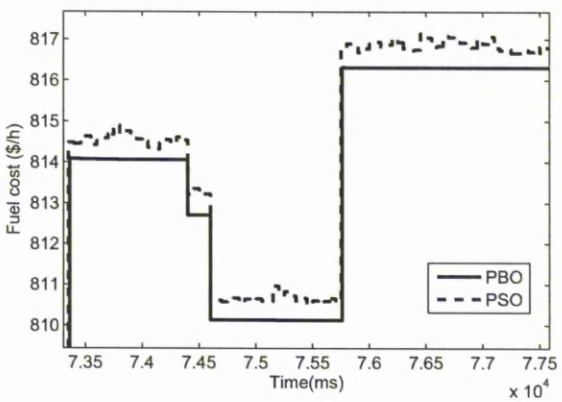


Figure 5.6: The detailed fuel cost of the system operated by the parameters from PBO and PSO ($\tau = 0.05$)

cost ($/h)		Best	Average	Standard Deviation
Case I	PBO	804.3789	805.4484	0.0987
	PSO	804.9361	805.8290	0.5682
Case II	PBO	793.0167	794.2588	0.1338
	PSO	793.9406	794.8434	0.8682
Case II	PBO	803.1470	804.6442	0.2645
	PSO	804.0126	805.1825	1.0048

Table 5.1: Fuel cost of the system operated by the parameters from PBO and PSO

The best result, average results and standard deviation of PBO and PSO from 30 runs are shown in Table 5.1. The comparisons of the above two algorithms are given in Figures 5.1~5.6. It is demonstrated in these figures that PBO is able to trace the environmental changes more effectively than PSO. The lack of population size causes a slow convergence in the earlier stages. However, the PBO is more accurate than PSO after convergence. In contrast, the individuals in PSO follow the global best in each generation, so the reflection of the environmental change is slower than that in PBO. As a result, the results obtained by PSO are not as stable as those of PBO.

5.2.4 Summary

Some of the existing EAs have been used in recent years to solve the dynamic OPF problems. However, most of them are too complex to adapt rapidly enough for the environmental changes, and few successful results have been reported. PBO was simulated on the dynamic environment of the OPF problems with three different levels of load changes in probability. Meanwhile, the performance of PBO was compared with PSO in the experimental studies. The experimental results have shown that in all three ranges of environmental changes, PBO is able to provide satisfactory performance, and can trace most of the environmental changes rapidly.

5.3 Optimal Stochastic Power Flow

5.3.1 Background

PBO was applied to optimise the distribution of the fuel cost in the stochastic power system test case in this section. In past research, OPF has been widely studied using Evolutionary Algorithms (EAs), such as PSO [37] and Bacterial Foraging Algorithm with Varying Population (BFAVP) [88]. Most of the research assumes that the power system modelling is based on a stable environment, *i.e.*, the environment is constant during the optimisation. As shown in the research of [147], uncertainty also plays an important role in optimal dispatch, and OPF with uncertainties have been given increasing attention. Although some research used a predefined dynamic power load to simulate the power system environment, it was still not accurate to describe the real-world power system [148]. In order to provide a valuable reference for future power dispatching, a stochastic OPF model is introduced in this section. In the stochastic OPF, load uncertainty means the system operators cannot forecast the real load demand at load buses in the future, but the possible distribution of the load is known based on the statistical data of system operation. The objective of the optimisation problem is to dispatch the power supply subject to the minimum total fuel cost.

In order to evaluate the performance of PBO, it is applied to the optimisation of the stochastic power flow by estimating the fuel cost distribution function. The experimental results have shown that PBO not only estimates a reliable distribution function with optimised mean fuel cost and standard deviation, but also has a minimal computation time.

Section 5.3.2 gives the formulations of the OPF improved with a stochastic load. Then the experiment on the IEEE 30-bus system is presented in Section 5.3.3. The fuel cost distribution model estimated by Genetic Algorithm (GA), PSO, and PBO are compared after the experiment.

5.3.2 Optimal power flow with stochastic loads

Because of the uncertainty of the real power load, it is impossible to forecast the exact real load demand at load buses. As a result, it is necessary to generate a set of samples based on the distribution of the load. In this case, the power network parameters are assumed to be constant. To describe a stochastic OPF in an uncertain environment, the real power loads are randomly set using the Gaussian distribution. Denote the real power load of the i^{th} bus by $\hat{P}_{\text{D}_i}$. Hence, during each power flow evaluation process, $\hat{P}_{\text{D}_i}$ is generated as a set of random scalars and is expressed as:

$$\hat{P}_{\text{D}_i} = \left[\hat{P}_{\text{D}_i,1} \quad \hat{P}_{\text{D}_i,2} \quad \dots \quad \hat{P}_{\text{D}_i,N_\text{S}}\right] \sim \mathcal{N}(\mu_{\text{D}_i}, \sigma_{\text{D}_i}), \tag{5.3.1}$$

where N_S indicates the number of the samples in a power flow evaluation, which is the same for all the buses, μ_{D_i} is the mean value, and σ_{D_i} is the standard deviation.

5.3.3 Experimental studies and results

In this experiment, PBO was applied to the standard IEEE 30-bus test case [4] to estimate the optimal fuel cost distribution functions. To estimate each distribution function, a number of samples of fuel cost were generated in one power flow evaluation with randomly distributed real power loads. The fuel cost of the j^{th} sample $\bar{F}_{\text{cost}_j}$ is calculated by the following equations:

$$\bar{F}_{\text{cost}_j} = \sum_{i=1}^{N_\text{G}} f_{\text{cost}_{i,j}}, \tag{5.3.2}$$

$$f_{\text{cost}_{i,j}} = a_i + b_i P_{\text{G}_{i,j}} + c_i P_{\text{G}_{i,j}}^2, \tag{5.3.3}$$

where $f_{\text{cost}_{i,j}}$ indicates the fuel cost (\$/h) of the i^{th} generator in the j^{th} sample, a_i, b_i, and c_i are fuel cost coefficients, and $P_{\text{G}_{i,j}}$ is the real power output generated by the i^{th} generator in j^{th} sample. The optimisation aims to minimise the mean and standard deviation of the fuel cost, which is expressed as follows:

$$F = \lambda_{\text{mean}} \cdot F_{\text{mean}} + \lambda_{\text{std}} \cdot F_{\text{std}}, \tag{5.3.4}$$

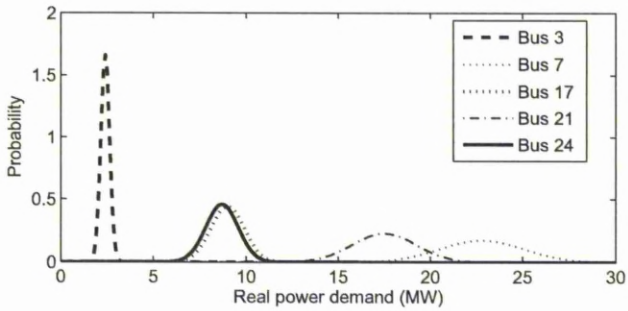


Figure 5.7: The distribution of the real power demands

where

$$F_{\text{mean}} = \sum_{j=1}^{N_{\text{A}}} \frac{\bar{F}_{\text{cost}_j}}{N_A} \tag{5.3.5}$$

$$F_{\text{std}} = \sqrt{\frac{\sum_{j=1}^{N_{\text{A}}} (\bar{F}_{\text{cost}_j} - \sum_{k=1}^{N_{\text{A}}} \bar{F}_{\text{cost}_k}/N_{\text{A}})^2}{N_{\text{A}}}} \tag{5.3.6}$$

Inequations (5.3.4), (5.3.5) and (5.3.6), λ_{mean} indicates the coefficient of the mean fuel cost, F_{mean} indicates the mean fuel cost of the evaluation, which is calculated from equation (5.3.5), λ_{std} indicates the coefficient of the standard deviation of the fuel cost, F_{std} indicates the standard deviation of fuel cost, which is calculated from equation (5.3.5), and N_{A} indicates the number of samples in each evaluation.

The objective function of the stochastic OPF is more expensive than that of the static OPF due to the computation complexity of sample evaluation of each F_{cost_j}. In this simulation, λ_{mean} and λ_{std} were set to 1 and 100, respectively, due to their feasible ranges. Meanwhile, N_{A} is set to 50, *i.e.*, 50 samples with different real power loads were taken in each evaluation.

The layout of the IEEE 30-bus test case is illustrated in Figure 4.2. In each evaluation process, the real power loads of nodes 3, 7, 17, 21, and 24 are the stochastic variables given with a Gaussian distribution with a mean value of the original rated value and a standard deviation of 10% of the mean value. The distribution of the real power demand is illustrated in Figure 5.7.

The performance of PBO is compared with the performances of GA and PSO in this experiment. The parameters of the algorithms are the same as described in Section 3.3. The mean and standard deviation of the fuel cost distribution function estimated by GA, PSO, and PBO are listed in Table 5.2, which are the average of 30 runs. In order to test their veracity, a two-sample t-test was carried out on these distribution functions. The fuel cost distribution functions estimated by the three algorithms were assumed as hypothesis models, and then 50 new samples were generated as the comparison data. The threshold chosen for statistical significance was set to 0.05 in the t-test. The p-values obtained in the t-tests are listed in Table 5.2 as well.

	F_{mean} ($/h)	F_{std} ($/h)	p-value	CPU time (h)
GA	805.8491	7.2767	0.6544	2.71
PSO	804.5307	5.9873	0.5076	2.84
PBO	804.1164	5.4122	0.7691	2.25

Table 5.2: The average mean and standard deviation of the fuel cost model estimated by GA, PSO, and PBO, t-test results and computation time

As shown by the t-test results; all three algorithms successfully described the fuel cost model with an acceptable error. Among them, PBO has the minimal mean fuel cost and standard deviation. At detailed stochastic comparison is illustrated in Figure 5.8, and it can be seen that the fuel cost model estimated by PBO is more stable than in the other two models. Meanwhile, the computation load of PBO is less than that of GA and PSO. Because the major computation in GA is logical computation, the time consumption of GA is slightly less than that of PSO.

5.3.4 Summary

This section described the stochastic OPF problem, which involves more uncertainties in the power system voltage control and generation dispatch. In

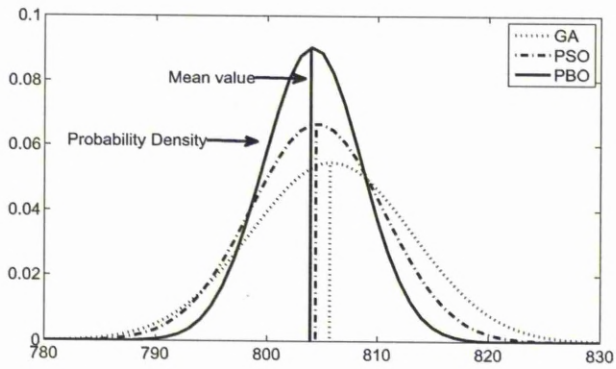


Figure 5.8: Probability density of the fuel cost model estimated by GA, PSO, and PBO

a stochastic OPF, the real power loads were represented by a set of random scalars with Gaussian distribution. Although the stochastic model is considered, the computational complexity increased accordingly. In order to solve this expensive optimisation problem, PBO was developed. PBO was applied to minimise the mean and the standard deviation of the fuel cost in the stochastic OPF. The experimental results have shown that PBO could effectively reduce the mean and standard deviation of the fuel cost in stochastic power system environments with an outstanding convergence speed.

5.4 Conclusion

In this Chapter, PBO was employed to solve two power system optimisation problems. PBO is a simple EA, which only contains two bacteria in a population. Two searching principles, both based on the distance between the two bacteria, are implemented. As a result, the gradient searching and the random searching are combined in PBO. In the first application, a novel approach to dynamic OPF problems has been presented. The experimental results show that PBO is able to trace the variation of fuel cost rapidly. Then PBO is applied to optimise the fuel cost distribution in a power system with stochastic

loads. Compared with GA and PSO, the computational complexity of PBO is significantly less. Moreover, thanks to its population diversity, PBO is suitable for applications in both dynamic environments and stochastic environments.

Chapter 6

Conclusion

This chapter concludes the thesis and summarises the achievements of the work done on bacteria-inspired optimisation algorithms and their applications. Suggestions for future research are also included.

6.1 Summary of Results

This research has been undertaken with the aims to develop bacteria-inspired optimisation algorithms by adopting the knowledge obtained from the studies of bacterial behaviours and apply them for power system optimisation. In this thesis, two novel optimisation algorithms, BFAVP and PBO, have been developed. Following the methodological studies, the developed algorithms have been applied to resolve the power system economic dispatch, voltage control, and harmonics problems. The summary of the research work presented in this thesis is listed below.

Chapter 1 began with the explanation of the motivations and objectives of this work. Various EAs were then reviewed, including GA, EP, GP, ES, PSO, ACO, GSO, and BFA. Meanwhile, the background of bacterial foraging behaviours was introduced. The major contributions of this thesis were presented at the end of the chapter.

The first part of the thesis was devoted to the development of the two novel BIAs, BFAVP and PBO. In Chapter 2, the bacterial foraging patterns and

mathematical framework of BFAVP were investigated in detail, which include the studies of modelling chemotaxis, metabolism, proliferation and elimination, and quorum sensing. Once the further understanding of bacterial foraging behaviours was achieved, the mathematical equations were developed to form the BFAVP algorithm. The BFAVP has been evaluated on a large set of benchmark functions that cover almost all cases of complex optimisation problems, in comparison with the popularly used EAs . From the simulation results obtained, It has been found that the performance of BFAVP has the performance similar to the other EAs for unimodal functions. However, the algorithm greatly outperforms the other EAs on most of the multimodal functions. Then BFAVP has been evaluated the experimental studies, with a range of algorithm parameters, to investigate the sensitivities of the performance to the algorithm performance. Finally, in order to demonstrate the advantages of the varying population size framework, a series of experiments have been undertaken. The experimental results show that BFAVP has not only an excellent convergence speed but is also able to achieve global search with finding local optima in a single run.

Chapter 3 has described another bacteria-inspired optimisation algorithm, PBO. The algorithm was developed with the aim of enhancing computational efficiency. Following this target, PBO have been developed with a simple structure, which contains only two individuals in a population. In each iteration, these two individials perform gradient searching and random searching alternately. In this chapter, the PBO has been evaluated on the benchmark functions adopted in Chapter 2. The experimental studies have shown that PBO is able to achieve superior results on multimodal functions with a much smaller number of function evaluations, in comparison with other EAs, and the better search performance in terms of accuracy and convergence speed. Furthermore, the number of mathematical and logical operations of PBO has been calculated to demonstrate its lower computational load in comparison with the other EAs.

The second part of this thesis has devoted to power system applications. In order to evaluate the proposed algorithms, the research work initially focussed

on the setup of comprehensive experimental environments, which includes following five aspects: 1) The power system simulation software packages which are used to calculate the power flow and fuel costs with constraints applied to the bus, branch and generator data under different conditions of power system operation. 2) A power system partitioning method which is developed to divide the IEEE 118-bus test case into 2 independent test cases by mimicking the rest of the system as a dummy generators. 3) Four models of different types of FACTS devices which are used in a power system to control the power flow and reduce the real power loss. 4) A power system simulation program which generates supposititious harmonic pollution with consideration of the variations of fundamental frequency, phases and amplitudes of integral-harmonics and inter-harmonics. 5) A statistical model and its program which estimates the probability distribution from a sequence of fuel costs generated by the power flow program, with providing stochastic loads. The experimental environments addressed above provide various scenarios for the comparison between proposed BIAs and other EAs.

Chapter 4 began with a brief introduction to four power system applications: OPF, partitioned OPF, allocation of FACTS devices, and estimation of harmonic parameters. Compared with GA and PSO, BFAVP is able to significantly reduce the fuel costs for the OPF computation of the IEEE 30-bus and IEEE 118-bus test systems respectively. Meanwhile, BFAVP also outperforms GA and PSO in reducing the fuel cost in the partitioned OPF. With dividing the complex IEEE 118-bus system into two partitions, BFAVP is able to optimise each partition simultaneously, and the time consumption of the optimisation process has been greatly reduced. In the third application, using FACTS devices to reduce the real power loss has been investigated. BFAVP was employed to meet the demand of multi-objective optimisation and to optimise the locations of FACTS devices, including their types and ratings. The application of BFAVP for optimal FACTS locations and control has been evaluated in the IEEE 14-bus and IEEE 30-bus test systems respectively. With the optimal location of these FACTS devices, found by BFAVP optimising the fuel

cost, the voltage profile was also improved. In the last application, two deviated harmonic fundamental frequencies with some variation were successfully estimated by BFAVP. Compared with GA and DFT, the harmonic parameters estimated by BFAVP have exhibited minimal error associated with the reconstructed signal. These four applications have verified that BFAVP is a competent algorithm for power system optimisation problems.

Chapter 5 demonstrated two applications of PBO for power system optimisation: solving OPF problems in a dynamic environment, and optimising fuel cost with stochastic real power loads. In the first application, a brief introduction to the OPF problem which contains dynamic real power loads was given, followed by the experiment undertaken on the IEEE 30-bus test system with three different rates of environmental changes. PBO is able to rapidly trace the real-time minimal fuel cost, due to its simple structure. Then PBO was applied to estimate the optimal fuel cost distribution in the IEEE 30-bus test system with stochastic real power loads applied. The simulation results of these two applications show that PBO outperforms GA and PSO in terms of accuracy and convergence speed.

Conclusively, from the successful developments of the two novel EAs, this thesis demonstrates the prospects of the studies of bacterial foraging behaviours. The knowledge of bacterial behaviours provides a new vision for high-dimensional optimisation problems. This thesis also demonstrates the outstanding performance of these two algorithms while they are applied to solve power system optimisation problems.

6.2 Suggestions for Future Work

In this section, suggestions for future work that aims to develop and improve the algorithms and applications are listed.

1. As discussion in Chapter 1, the most important feature that distinguishes BFA from other EAs is the reproduction and elimination in bacterial behaviours. The varying population framework in BFAVP is developed from

these two features. The experiments in Chapter 2 also demonstrate that the varying population framework not only accelerates the convergence speed of BFAVP but also increases the population diversity. However, these features are not sufficiently discovered to improve the performance of other EAs. Thus, further work needs to be done to improve EAs based on inspiration from the varying population framework.

2. The experiments in Chapter 2 show that although BFAVP has made noticeable progress on multimodal functions, its performance on unimodal functions is still not greatly improved. To overcome this drawback, PBO in Chapter 3 adopts gradient searching as the major searching method. However, the pseudo gradient is only approximate to the real gradient value. It is worthwhile to enhance the performance on multimodal functions by investigating the gradient searching method.

3. Chapter 3 shows that time consumption of PBO is much less than other EAs. The convergence speed is accelerated by simplified quorum sending. Simplified quorum sensing exchanges the gradient information among individuals, while congregating individuals to the global optima. It is worthy to study this behaviour and introduce the concept to other EAs in further work.

4. This study extends the application of BIAs in engineering problems, especially in power systems. However, the gap between the modelling of biological systems and the applications to real-world problems requires a large amount of work in this area. In further work, more large-scale real-world problems can be solved by applying the algorithms proposed in this research.

Appendix A

Benchmark Functions

These benchmark functions are employed from [9].

A.1 High-dimensional Unimodal Benchmark Functions

Sphere Model

$$f_1(x) = \sum_{i=1}^{30} x_i^2 \qquad -100 \leq x_i \leq 100$$
$$\min(f_1) = f_1(0, ..., 0) = 0$$

Schwefel's Problem 2.22

$$f_2(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i| \qquad -10 \leq x_i \leq 10$$
$$\min(f_2) = f_2(0, ..., 0) = 0$$

Schwefel's Problem 2.21

$$f_3(x) = \max_i \{|x_i|, 1 < i < 30\} \qquad -100 \leq x_i \leq 100$$
$$\min(f_3) = f_3(0, ..., 0) = 0$$

Generalised Rosenbrock's Function

$$f_4(x) = \sum_{i=1}^{29} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)]^2; \qquad -30 \leq x_i \leq 30$$

$$\min(f_4) = f_4(1, ..., 1) = 0$$

Step Function

$$f_5(x) = \sum_{i=1}^{30}(\lfloor x_i + 0.5 \rfloor)^2 \quad -100 \leq x_i \leq 100$$
$$\min(f_5) = f_5(0, ..., 0) = 0$$

A.2 High-dimensional Multimodal Benchmark Functions

Generalised Schwefel's Problem 2.26

$$f_6(x) = \sum_{i=1}^{30} -x_i \sin(\sqrt{|x|}) \quad -500 \leq x_i \leq 500$$
$$\min(f_6) = f_6(420.9687, ..., 420.9687) = -12569.5$$

Generalised Rastrigin's Function

$$f_7(x) = \sum_{i=1}^{30}[x_i^2 - 10\cos(2\pi x_i) + 10]^2 \quad -5.12 \leq x_i \leq 5.12$$
$$\min(f_7) = f_7(0, ..., 0) = 0$$

Ackley's Function

$$f_8(x) = -20\exp\Big(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{30}x_i^2}\Big) - \exp\Big(\frac{1}{30}\sum_{i=1}^{30}\cos(2\pi x_i)\Big) + 20 + e$$
$$-32 \leq x_i \leq 32$$
$$\min(f_8) = f_8(0, ..., 0) = 0$$

Generalised Girewank Function

$$f_9(x) = \frac{1}{4000}\sum_{i=1}^{30}x_i^2 - \prod_{i=1}^{30}\cos\Big(\frac{x_i}{\sqrt{i}}\Big) + 1$$
$$-600 \leq x_i \leq 600$$
$$\min(f_9) = f_9(0, ..., 0) = 0$$

Generalised Penalised Function

$$
\begin{aligned}
f_{10}(x) &= \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{29}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_{29})^2\} \\
&\quad + \sum_{i=1}^{30} u(x_i, 5, 100, 4), \\
&\quad -50 \le x_i \le 50 \\
&\quad \min(f_{10}) = f_{10}(1, ..., 1) = 0 \\
\text{where} \quad &\quad y_i = 1 + \frac{1}{4}(x_i + 1), \\
&\quad u(x_i, a, k, m) = \begin{cases} k(x_i-1)^m, & x_i > a, \\ 0, & -a < x_i < a, \\ k(-x_i-1)^m, & x_i < -a. \end{cases}
\end{aligned}
$$

A.3 Low-dimensional Multimodal Benchmark Functions

Shekel's Foxhole Function

$$
\begin{aligned}
& f_{11}(x) = \Big[\frac{1}{500} + \sum_{j=1}^{2} 5 \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\Big] \\
& -65.536 \le x_i \le 65.536 \\
& \min(f_{11}) \approx f_{11}(-32, -32) \approx 1 \\
\text{where} \quad & (a_{ij}) = \begin{pmatrix} 32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{pmatrix}
\end{aligned}
$$

Kowalik's Function

$$
\begin{aligned}
& f_{12}(x) = \sum_{i=1}^{11} \Big[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\Big]^2 \quad -6 \le x_i \le 6 \\
& \min(f_{12}) \approx f_{12}(0.1928, 0.1928, 0.1231, 0.1358) \approx 0.0003075
\end{aligned}
$$

Six-Hump Camel-Back Function

$$
\begin{aligned}
& f_{13}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \\
& -5 \le x_i \le 5 \\
& \min(f_{13}) = f_{13}(0.08983, -0.7126) = -1.0316285
\end{aligned}
$$

Table A.1: Kowalk's function f_{12}

i	1	2	3	4	5	6
a_i	0.1957	0.1947	0.1735	0.1600	0.0844	0.0627
b_i^{-1}	0.25	0.5	1	2	4	6
i	7	8	9	10	11	
a_i	0.0456	0.0342	0.0323	0.235	0.0246	
b_i^{-1}	8	10	12	14	16	

A.4 Dynamic Functions

Quartic Function with Noise

$$f_{14}(x) = \sum_{i=1}^{30} ix_i^4 + \text{random}[0,1) \qquad -1.28 \leq x_i \leq 1.28$$
$$\min(f_{14}) = f_{14}(0,...,0) = 0$$

Generalised Schwefel's Problem 2.26 with Noise

$$f_{15}(x) = \sum_{i=1}^{30} -x_i \sin(\sqrt{|x|}) + \text{random}[0,1) \qquad -500 \leq x_i \leq 500$$
$$\min(f_{15}) = f_{15}(420.9687,...,420.9687) = -12569.5$$

Goldstein-Price Function

$$\begin{aligned} f_{16}(x) = {} & [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ & \times[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \\ & + \text{random}[0,1) \\ & -2 \leq x_i \leq 2 \quad \min(f_{14}) = f_{16}(0,-1) = 3 \end{aligned}$$

Appendix B

Notations in Thesis

B.1 Notations in BFAVP

X_p^k	the position of the p^{th} bacterium, in the tumble-run process of the k^{th} iteration
$F(X_p^k)$	the evaluation value of the bacterium
φ_p^k	the heading angle of the bacterium
$D_p^k(\varphi_p^k)$	the tumble length of the bacterium
$\varphi_{\max}$	the maximal rotation angle
$\hat{\varphi}_p^k$	the head angle of the bacterium after tumble
N_{c}	the maximal number of run steps
$\hat{X}_p^{k,h}$	the position of the bacterium at the h^{th} run step
$D_{\max}$	the maximal step length
N_{f}	the number of the run steps performed
e_p^k	the energy of the bacterium at the beginning of the k^{th} iteration
α	the coefficient for energy transform
$F_{\min}$	the minimal evaluation value in the history
$\tilde{e}_p^k$	the energy after the tumble-run process
O_p^k	the age of the bacterium
μ_{span}	the mean of the bacterial lifespan
σ_{span}	the standard deviation of the bacterial lifespan

$\mathcal{P}$	the probability density of bacterial lifespan expectancy
X_{best}	the position of the current best global solution updated after the evaluation of each function
ζ	the rate of bacteria performed repelling
D_{range}	the range of the search space

B.2 Notations in PBO

X^k	the position of the primary individual at the k^{th} iteration
$\widetilde{X}^k$	the position of the pseudo individual at the k^{th} iteration
δD^k	the distance between the primary individual and pseudo individual
d_l^k	the distance between the primary individual and pseudo individual on the l^{th} dimension
B_{lo_l}	the lower boundary on the l^{th} dimension
B_{up_l}	the upper boundary on the l^{th} dimension
c_{c}	the two-value coefficient for placing the pseudo individual
$g_l^k(\widetilde{X}^k, X^k)$	the pseudo gradient between the primary individual and pseudo individual
Vel_{c}^k	velocity of the chemotaxis in the k^{th} iteration
Vel^k	the velocity of the primary individual
γ	the inertia weight of the velocity
P_{g}^k .	the position of the best individual form the past k iterations
$vel_l^{\max}$	the maximal velocity of primary individual along the l^{th} dimension
Vel_{q}^k	the velocity of attraction in the k^{th} iteration
Vel_{s}^k	the velocity of learning in the k^{th} iteration

B.3 Notations in Power System

θ_{ij}	voltage angle difference between bus i and j (rad)
B_{ij}	transfer susceptance between bus i and j (p.u.)
G_{ij}	transfer conductance between bus i and j (p.u.)
g_k	conductance of branch k (p.u.)
N_0	the total number of total buses excluding slack bus
N_{B}	the total number of total buses
N_{C}	the total number of shunt compensators
N_{D}	the total number of power demand buses
N_{E}	the total number of network branches
N_{G}	the total number of generator buses
N_i	the total number of buses adjacent to bus i, including bus i
N_{PQ}	the total number of PQ buses
N_{PV}	the total number of PV buses
$N_{\mathrm{Q}}^{\mathrm{lim}}$	the total number on buses on which injected reactive power outside limits
N_{T}	the total number of transformer branches
N_V^{lim}	the total number on buses on which voltages outside limits
P_{D_i}	the demanded active power at bus i (p.u.)
P_{G_i}	the injected active power at bus i (p.u,)
Q_{C_i}	the reactive power source installation at bus i (p.u.)
Q_{D_i}	the demanded reactive at bus i (p.u.)
Q_{G_i}	the injected reactive power at bus i (p.u.)
V_{G}	the voltage vector of PQ buses (p.u.)
T_i	the tap position at transformer i
V_i	the voltage magnitude at bus i (p.u.)
S_k	the apparent power flow in branch k (p.u.)
u_{d}	the vector of dependent variables such

u_c	the vector of dependent variables such
F_{cost}	the fuel cost of the system
λ_{V_i}	the penalty factors of the voltage on the i^{th} bus
λ_{G_i}	the penalty factors of the reactive power on the i^{th} bus
λ_{S_i}	the penalty factors of the apparent power flow on the i^{th} bus
f_{cost_i}	the fuel cost of the i^{th} generator
N_{P}	the number of partitions in partitioned OPF
$\triangle Q_i$	the injected power of the SVC at bus i
E_{ij}	the reactance of line ij
C_{F}	the cost of FACTS devices
P_{max}	the thermal limit of the branch where the TCPST is to be installed
IC	the installation cost of the TCPST
C_{FH}	the average investment cost of installation of the FACTS devices
Q_{SVC}	the reactive power injected or absorbed by the SVC
E_{TCSC}	the value of the capacitance or inductance of the TCSC
θ_{TCPST}	the angle of bus voltage, adjusted by the TCPST
T_{UPFC}	the transformer voltage magnitude turns ratio of UPFC
ϕ_{UPFC}	the phase shifting angle of UPFC
Q_{UPFC}	the reactive power injected or absorbed by the UPFC
J_{SC}	the system security constraint
Vl_i	the voltage levels of bus i
V_{L_i}	the voltage magnitude at bus i
Bol_j	the branch loading and penalty overloads in the bus j
λ	the system load parameter

References

[1] J. W. Bell. *Searching Behaviour - The Behavioural Ecology of Finding Resources,* Chapman and Hall Animal Behaviour Series. Chapman and Hall, 1990.

[2] N. A. Campbell and J. B. Reece. *Biology: International Edition.* Benjamin Cummings, 1301 Sansome St., San Francisco, CA 94111, USA, 2005.

[3] U.S. Department of Energy. *Final Report on the August 14, 2003 Blackout in the United States and Canada.* U.S.-Canada Power System Outage Task Force, 2006.

[4] O. Alsac and B. Scott. Optimal load flow with steady state security. *IEEE Trans. on Power Appara. Syst.*, PAS-93:745–751, May-June 1974.

[5] Q. H. Wu, T. Y. Ji, M. S. Li, and Z. Lu. Distributed optimal power flow using bacterial swarming algorithm. *International Conference on Modelling, Identification and Control*, pages 1–6, 2008.

[6] Z. Lu, M. S. Li, W. J. Tang, and Q. H. Wu. Multi-objective optimization of reactive power dispatch using a bacterial swarming algorithm. *Chinese Control Conference*, pages 460 – 464, 2007.

[7] P. K. Iyambo and R. Tzoneva. Transient stability analysis of the ieee 14-bus electric power system. *AFRICON*, pages 1–9, 2007.

[8] W. C. Davidon. Variable metric method for minimization. *SIAM J. Optim.*, 1(1):1–17, 1991.

[9] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3:82–102, July 1999.

[10] W. J. Tahng. *Optimisation algorithms inspired from modelling of bacterial foraging patterns and their applications.* Ph.D. thesis, Liverpool University, Liverpool, UK, 2008.

[11] C. H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1995.

[12] S. He, E. Prempain, and Q. H. Wu. An improved particle swarm optimiser for mechanical design optimisation problems. *Engineering Optimisation*, 36(5):585–605, 2004.

[13] W. H. Tang, S. He, E. Prempain, Q. H. Wu, and J. Fitch. A particle swarm optimiser with passive congregation approach to thermal modelling for power transformers. *2005 IEEE congress on Evolutionary Computation (CEC2005),* Edinbutgh, September 2005.

[14] C. A. Brebbia. *Biologically Inspired Optimization Methods: An Introduction.* WIT Press, Ashurst Lodge, Southampton, SO40 7AA, UK, 2008.

[15] B. Gartner and Jiri Matousek. *Understanding and Using Linear Programming.* Springer, Berlin, German, 2006.

[16] M. Avriel. *Nonlinear Programming: Analysis and Methods.* Dover Publishing, Mineola, NY, USA, 2003.

[17] R. T. Rockafellar. *Convex Analysis. Princeton University Press.* Princeton, NY, USA, 1970.

[18] D. G. Bounds. New optimization methods from physics and biology. *Nature*, 329:215–219, September 1987.

[19] S. He. *Developments of animal behaviour inspired optimisation algorithms and their applications.* Ph.D. thesis, Liverpool University, Liverpool, UK, 2006.

[20] X. S. Yang. *Nature-inspired meta-heuristic algorithms.* Luniver Press, 2008.

[21] J. Hertz, R. G. Palmer, and A. S. Krogh. *Introduction to the theory of neural computations.* Perseus Books, 1990.

[22] D. B. Fogel. The advantage of evolutionary computation. *Bio-Computing and Emergent Computation*, pages 1–11, 1997.

[23] H. Lipson and J. B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.

[24] K. Gurney. *An Introduction to Neural Networks.* Routledge, London, 1997.

[25] J. Holland. *Adaptation in nature and artificial systems.* The University of Michigan Press, Ann Arbor, Michigan, 1975.

[26] D. E Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning.* Addison Wesley, 1989.

[27] F. Herrera, M. Lozano, and J. L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artif. Intell. Rev.*, 12(4):265–319, 1998.

[28] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 1:69–93, 1991.

[29] L. Fogel. *Artificial intelligence through Simulated Evolution.* John Wiley and Sons Inc., New York, USA, 1966.

[30] J. R. Koza. *Genetic programming: on the programming of computers ny means of natural selection.* MIET Press, 1992.

[31] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone. *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications.* Morgan Kaufmann, 1998.

[32] I. Rechenberg. *Evolution strategie: optimieriung rechnischer systemenach prinzipien der biologichen evolution.* Frommann-Holzboog, Stuuugart, Germany, 1973.

[33] H. P. Schwefel. *Numerical Optimization of computer models.* Wiley, Chichester, 1981.

[34] T. Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* Oxford University Press, Oxford, UK, 1996.

[35] J. Kennedy and R.C. Eberhart. Particle swarm optimization. volume 4, pages 1942–1948, Perth, Australia, 1995.

[36] M. Clerc and J. Kenney. The particle swarm - explosion, stability, and convergence in a multimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[37] S. He, Q. H. Wu, J. Y. Wen, J. R. Saunders, and R. C. Paton. A particle swarm optimizer with passive congregation. *BioSystems*, 78:135–147, 2004.

[38] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimiser for global optimisation of multimodal functions. *IEEE Transactions on Evolutionary Compytation*, 10(3):281–295, 2006.

[39] A. Colorni, M. Dorigo et, and V. Maniezzo. *Distributed Optimization by Ant Colonies, actes de la premire confrence europenne sur la vie artificielle.* Elsevier Publishing, Paris, France, 1991.

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

[41] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence.* Oxford University Press, Oxford, UK, 1999.

[42] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Compytation*, 1(1):53–66, 1997.

[43] V. Maniezzo, L. M. Gambardella, and F. D. Luigi. Ant colony optimization. In *Optimization Techniques in Engineering. Springer-Verlag*, pages 101–117. Addison-Wesley, 2004.

[44] S. He, Q. H. Wu, and J. R. Saunders. A group search optimiser for neural network training. volume 3948, pages 1324–1330, Glasgow, UK, May 8-11 2006.

[45] C. W. Clark and M. Mangel. Foraging and flocking strategies: information in an uncertain environment. *Am. Nat.*, 123:626–641, 1984.

[46] C. J. Barnard and R. M. Sibly. Producers and scroungers: a general model and its application to captive flocks of house sparrows. *Animal Behaviour*, 29:543–550, 1981.

[47] W. J. OBrien, B. I. Evans, and G. L. Howick. A new view of the predation cycle of a planktivorous fish, white crappie (pomoxis annularis). *Can. J. Fish. Aquat. Sci.*, 43:1894–1899, 1986.

[48] K. M. Passino. Biomimicry of bacterial foraging for distributed optimisation and control. *IEEE Control Systems Magazine*, 22(3):52–67, June 2002.

[49] G. Lowe, M. Meister, and H. Berg. Rapid rotation of flagellar bundles in swimming bacteria. *Nature*, 325:637–640, 1987.

[50] H. Berg. *Random Walks in Biology.* Princeton Univ. Press, Princeton, NJ, 1993.

[51] T. M. Yi, Y. Huang, M. Simon, and J. Doyle. Robust perfect adaptation in bacterial chemotaxis through integral feedback control. *PNAS*, 97:4649–4653, 2000.

[52] A. Biswas, S. Dasgupta, S. Das, and A. Abraham. Synergy of pso and bacterial foraging optimisation: a comparative study on numerical benchmarks. *Second International Symposium on Hybrid Artificial Intelligent Systems*, pages 255–263, 2007.

[53] D. H. Kim and A. Abraham. A hybrid genetic algorithm and bacterial foraging approach for global optimization and robust tuning of pid controller with disturbance rejection. *Studies in Computational Intelligence*, 75:171–199, 2007.

[54] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, USA, 1999.

[55] T. Datta, I. S. Misra, B. B. Mangaraj, and S. Imtiaj. Improved adaptive bacteria foraging algorithm in optimisation of antenna array for faster convergence. *Progress In Electromagnetics Research*, 1:143–157, 2008.

[56] B. P Lathi. *Modern digital and analog communication systems*. Oxford University Press, Oxford, UK, 1998.

[57] W. J. Tang, Q. H. Wu, and J. R. Saunders. Bacterial foraging algorithm for dynamic environment. *in Proceeding of 2006 IEEE Congress on Evolutionary Computation (CEC2006)*, pages 16–21, July 2006.

[58] W. J. Tang, M.S. Li, S. He, Q. H. Wu, and J. R. Saunders. Optimal power flow with dynamic loads using bacterial foraging algorithm. *in Proceeding of 2006 International Conference on Power Systems Technology*, pages 22–26, October 2006.

[59] A. van Leeuwenhoek. An abstract of a letter from mr. anthony leevvenhoek at delft, dated sep. 17, 1683, containing some microscopical observations, about animals in the scurf of the teeth, the substance call'd worms

in the nose, the cuticula consisting of scales. *Philosophical Transactions*, 1683–1775(14):568–574, 1684.

[60] A. van Leeuwenhoek. Part of a letter from mr antony van leeuwenhoek, concerning the worms in sheeps livers, gnats, and animalcula in the excrements of frogs. *Philosophical Transactions*, 1683–1775(22):509–518, 1700.

[61] A. van Leeuwenhoek. Part of a letter from mr antony van leeuwenhoek, f. r. s. concerning green weeds growing in water, and some animalcula found about them. *Philosophical Transactions*, 1683–1775(23):1304–1311, 1702.

[62] J. R. Poter. Antony van leeuwenhoek: Tercentenary of his discovery of bacteria. *Bacteriological reviews*, 40(2):260–269, 1976.

[63] P. Feng, S. Weagant, and M. Grant. Enumeration of escherichia coli and the coliform bacteria. *Bacteriological Analytical Manual (8th edition)*, January 2007.

[64] H. E. Kubitschek. Cell volume increase in escherichia coli after shifts to richer media. *Journal of Bacteriology*, 172(1):94–101, January 1990.

[65] R. Lux and W. Shi. Chemotaxis-guided movements in bacteria. *Crit Rev Oral Biol Med*, 15(4):207–220, 2004.

[66] C. W. Bowers, F. Lau, and T. J. Silhavy. Secretion of lamb-lacz by the signal recognition particle pathway of escherichia coli. *Journal of Bacteriology*, 185:5697–5705, 2003.

[67] S. H. Larsen, J. Adler, J. J. Gargus, and R. W. Hogg. Chemomechanical coupling without atp: the source of energy for motility and chemotaxis in bacteria. *Proc. Natl. Acad. Sci.*, 71:1239–1243, 1974.

[68] W. Zillig. Comparative biochemistry of archaea and bacteria. *Current Opinion in Genetics & Development*, 1(4):544–551, 1991.

[69] S. Khan and R. M. Macnab. Proton chemical potential, proton electrical and bacterial motility. *Journal of Motility Biology*, 138:599–614, 1980.

[70] M. J. Maham, J. M. Slauch, and J. J. Mekalanos. Environmental regulation of virulence gene expression in *Eschichia*, *Salmonella*, and *Shigella* spp. *Escherichia coli and Salmonella*, 2:2803–2816, 1996.

[71] A. Koch. Control of the bacterial cell cycle by cytoplasmic growth. *Crit Rev Microbiol*, 28(1):61–77, 2002.

[72] R. Eagon. Pseudomonas natriegens, a marine bacterium with a generation time of less than 10 minutes. *J Bacteriol*, 83(4):736–737, 1962.

[73] C. Prats, D. Lopez, A. Giro, J. Ferrer, and J. Valls. Individual-based modelling of bacterial cultures to study the microscopic causes of the lag phase. *Journal of Theoretical Biology*, 241(4):939–953, August 2006.

[74] M Hecker and U Volker. General stress response of bacillus subtilis and other bacteria. *Adv Microb Physiol*, 44:35–91, 2001.

[75] B. M. Ahmer. Cell-to-cell signalling in escherichia coli and salmonella enterica. *Mol. Microbiol.*, 52(4):933–945, 2004.

[76] M. G. Surette, M. B. Miller, and B. L. Bassler. Quorum sensing in escherichia coli, salmonella typhimurium, and vibrio harveyi: a new family of genes responsible for autoinducer production. *Proc. Natl. Acad. Sci.*, 96(4):1639–1644, 1999.

[77] E. Bonabeau, M. Dorigo, and G. Theraulza. Inspiration for optimisation from social insect behaviour. *Nature*, 406:39–42, July 2000.

[78] D. Powell and M. M. Skolnick. Using genetic algorithms in engineering design optimization with nonlinear constraints. *Int. Conf. On Genetic Algorithms*, pages 424–431, 1993.

[79] G. Harik, E. Cantu-Paz, D. Goldberg, and B. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.

[80] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS-a genetic algorithm with varying population size. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 73–78, Orlando, FL, USA, June 27-29 1994.

[81] T. Back, A. E. Eiben, and N. A. L. van der Sart. An empirical study on GAs "without parameters". In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 315–324, London, UK, September 2000.

[82] A. E. Eiben, E. Marchiori, and V. A. Valko. Evolutionary algorithms with on-the-fly population size adjustment. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 41–50, Birmingham, UK, September 2004.

[83] G. Harik and F. G. Lobo. A parameter-less genetic algorithm. volume 1, pages 258–265, San Fransisco, USA, 1999. Morgan Kaufmann.

[84] F. G. Lobo and D. E. Goldberg. The parameter-less genetic algorithm in practicer. *Information Sciences*, 167:217–232, 2004.

[85] N. Hansen, S. D. Muller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary Computation*, 11(1):1–18, 2003.

[86] D. H. Kim and J. H. Cho. Biomimicry of bacterial foraging for distributed optimization and control. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 9(6):669–676, July 2005.

[87] W. J. Tang, Q. H. Wu, and J. R. Saunders. A novel model for bacterial foraging in varying environments. In *Computational Science and its*

Applications - ICCSA 2006, volume 3980/2006 of *Lecture Notes in Computer Science*, pages 556–565, Glasgow, UK, May 2006. Springer Berlin / Heidelberg.

[88] M. Clerc and J. Kennedy. Bacterial foraging algorithm with varying population for optimal power flow. *Lecture Notes in Computer Science*, 4448:58–73, 2007.

[89] H. C. Berg. Motile behavior of bacteria. *Physics Today*, 53(1):24–29, January 2000.

[90] J. Bar Tana, B. J. Howlett, and D. E. Koshland. Flagellar formation in *Escherichia coli* electron transport mutants. *Journal of Bacteriology*, 130(2):787–792, May 1977.

[91] M. B. Miller and B. L. Bassler. Quorum sensing in bacteria. *Annual Review of Microbiology*, 55:165–199, 2001.

[92] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[93] S. He, Q. H. Wu, and J. R. Saunders. Group search optimiser - an optimisation algorithm inspired by animal searching behavior. *accepted by IEEE Transactions on Evolutionary Computation*, 2008.

[94] S. Dasgupta, S. Das, A. Abraham, and A. Biswas. Adaptive computational chemotaxis in bacterial foraging optimization: an analysis. *IEEE Transactions on Evolutionary Computing*, 13(4):919–941, 2009.

[95] C. R. Houck, J. A. Joines, and M. G. Kay. A genetic algorithm for function optimization: A matlab implementation. *Technical Report NCSU-IE-TR-95-09*, 1995.

[96] T. S. P. Duque, K. Sastry, A. C. B. Delbem, and D.E. Goldberg. Evolutionary algorithm for large scale problems. *Seventh International Conference on Intelligent Systems Design and Applications, 2007. ISDA 2007*, pages 819–822, 2007.

[97] Z. Ji, J. R. Zhou, H. L. Liao, and Q. H. Wu. Requantization codebook design using particle-pair optimiser. *IEEE Congress on Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence)*, pages 1851–1855, 2008.

[98] H. H. Bauschke, C. H. Hamilto, M. S. Macklem, J. S. McMichael, and N. R. Swart. Recompression of jpeg images by requantization. *IEEE Transactions On Evolutionary Computation*, 12(7):843–849, 2003.

[99] H. A. Abbass, A. M. Bagirov, and J. Zhang. The discrete gradient evolutionary strategy method for global optimisation. *Congress on Evolutionary Computation*, 1:8–12, 2003.

[100] M. Bundzel and P. Sincak. Combining gradient and evolutionary approaches to the artificial neural networks training according to principles of support vector machines. *IEEE International Joint Conference on Neural Networks*, pages 2068–2074, 2006.

[101] X. Hu, Z. Huang, and Z. Wang. Hybridisation of the multi-objective evolutionary algorithms and the gradient-based algorithms. *Congress on Evolutionary Computation*, 2:8–12, 2003.

[102] J. Hewlett, B. Wilamowski, and G. Dundar. Merge of evolutionary computation with gradient based method for optimisation problems. *IEEE International Symposium on Industrial Electronics*, pages 3304–3309, 2007.

[103] M. Manic and B. Wilamowski. Random weights search in compressed neural networks using over-determined pseudo-inverse. *IEEE Interantional Symposium on Industrial Electronics*, pages 678 – 683, 2003.

[104] M. Manic and B. Wilamowski. Robust neural network training using partial gradient probing. *IEEE International Conference on Industrial Informatics*, pages 175–180, 2003.

[105] J. Y. Wen, Q. H. Wu, L. Jiang, and S. J. Cheng. Pseudo-gradient based evolutionary programming. *Electronics Letters*, 39:631–632, 2003.

[106] B. Vatchova. Application of genetic algorithms by means of pseudo gradient. *Computational Intelligence, Theory and Applications*, 33(7):17–24, 2005.

[107] T. J. Ypma. Historical development of the newton-raphson method. *SIAM Review*, 37:531–551, 1995.

[108] Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985–2999, 2008.

[109] K. Chellapilla. Combining mutation operators in evolutionary programming. *IEEE Trans. on Evolutionary Computation*, 2:91–96, 1998.

[110] D. B. Fogel. *System Identification Through Simulated Evolution: A machine Learning Approach to modelling.* Ginn Press, 160 Gould Street, Needham Heights, MA 01294, 1991.

[111] Z. Zheng and C. Wang. Measures of suppressing harmonic pollution generated by power electronic equipment. *Power and Energy Engineering Conference*, pages 1–4, 2009.

[112] J. Carpentier. *Contribution to the econimoc dispatch problem*, volume 8. Bull. Soc. Franc. Elect, 1962.

[113] J. A. Momoh, R. J. Koessler, M. S. Bond, B. Scott, D. Sun, A. Papalexopoulos, and P. Ristanovic. Challenges to optimal power flow. *IEEE Transocean on Power Systems*, 12:444–455, 1997.

[114] M. H. Bottero, F. D. Galiana, and A. R. Fahmideh-Vojdani. Economic dispatch using the reduced hessian. *IEEE Trans. on Power Appara. Syst.*, PAS-101:3679–3688, Oct. 1982.

[115] G. F. Reid and L. Hasdorf. Economic dispatch using quadratic programming. *IEEE Trans. on Power Appara. Syst.*, PAS-92:2015–2023, 1973.

[116] B. Stott and E. Hobson. Power system security control calculation using linear programming. *IEEE Trans. on Power Appara. Syst.*, PAS-97:1713–1731, 1978.

[117] J. A. Momoh and J. Z. Zhu. Improved interior point method for opf problems. *IEEE Trans on Power Syst.*, 14:1114–1120, 1999.

[118] H. Wei, H. Sasaki, J. Kubokawa, and R. Yokoyama. An interior point nonlinear programming for optimal power flow problems with a novel structure. *IEEE Trans on Power Syst.*, 13:870–877, 1998.

[119] A. G. Bakirtzis, P. N. Biskas, C. E. Zoumas, and V. Petridis. Optimal power flow by enhanced genetic algorithm. *IEEE Transactions on Power Systems*, 17(2):229–236, MAY 2002.

[120] J. Grainger and W. Stevenson. *Power System Analysis.* McGraw-Hill, New York, 1994.

[121] S. He, J. Y. Wen, E. Prempain, Q. H. Wu, J. Fitch, and S. Mann. An improved particle swarm optimiser for optimal power flow. *International Conference on Power System Technology*, 2:1633–1637, 2004.

[122] R. Baldick, B. H. Kim, C. Chase, and Y. Luo. A fast distributed implementation of optimal power flow. *IEEE Transactions on Power Systems*, 14(3):858–864, 1999.

[123] A. J. Conejo and J. A. Aguado. Multi-area coordinated decentralized dc optihial power flow. *IEEE transactions on power systems*, 13:1272–1278, November 1998.

[124] P. N. Biskas and A. G. Bakirtzis. Decentralised opf of large multiarea power systems. *IEE Proceedings of Generation, Transmission and Distribution*, 153:99–105, Janurary 2006.

[125] B. H. Kim and R. Baldick. A comparison of distributed optimal power flow algorithms. *IEEE transactions on power systems*, 15(2):599–604, May 2000.

[126] E. Acha, C. R. Fuerte-Esquivel, H. Ambriz-Perez, and C. Angeles-Camacho. *FACTS: Modelling and Simulation in Power Networks.* Wiley, West Sussex, U.K., 2005.

[127] N. G. Hingorani and L. Gyugyi. *Understanding FACTS: Concepts and Technology of Flexible AC Transmission Systems.* IEEE Press, Piscataway, NJ, 1999.

[128] W. Shao and V. Vittal. LP-based OPF for corrective FACTS control to relieve overloads and voltage violations. *IEEE Transactions on Power Systems*, 21(4):1832–1839, December 2006.

[129] S. Gerbex, R. Cherkaoui, and A. J. Germond. Optimal location of multitype facts devices in a power system by means of genetic algorithms. *IEEE Transactions on Power Systems*, 16(3):537C544, Augest 2001.

[130] M. Saravanan, S. M. R. Slochanal, P. Venkatesh, and J. P. S. Abraham. Application of particle swarm optimization technique for optimal location of facts devices considering cost of installation and system loadability. *Electric Power Systems Research*, 77:276C283, 2007.

[131] L. J. Cai and I. Erlich. Simultaneous coordinated tuning of pss and facts damping controllers in large power systems. *IEEE Transactions on Power Systems*, 20(1):294C300, 2005.

[132] Z. Lu, M. S. Li W. J. Tang, and Q. H. Wu. Optimal location of facts devices by a bacterial swarming algorithm for reactive power planning. *In Proceedings of 2007 IEEE Congress on Evolutionary Computation (CEC 2007),* Singapore, 20(1):2344C2349, September 25-28 2007.

[133] R. Mínguez, F. Milano, R. Zárate-Miñano, and A. J. Conejo. Optimal network placement of SVC devices. *IEEE Transactions on Power Systems*, 22(4):1851–1860, Nov. 2007.

[134] J. Zhang, J. Y. Wen, S. J. Cheng, and J. Ma. A novel svc allocation method for power system voltage stability enhancement by normal forms of diffeomorphism. *IEEE Transactions on Power Systems*, 22(4):1819–1825, Nov. 2007.

[135] Z. Cai, L. Zhu, Z. Lan, D. Gan, Y. Ni, L. Shi, and T. Bi. A study on robust adaptive modulation controller for TCSC based on COI signal in interconnected power systems. *Electric Power Systems Research*, 78(1):147–157, Jan. 2008.

[136] S. An, J. Condren, and T. W. Gedra. An ideal transformer upfc model, opf first-order sensitivities, and application to screening for optimal UPFC locations. *IEEE Transactions on Power Systems*, 22(1):68–75, Feb. 2007.

[137] L.J. Cai and I. Erlich. Optimal choice and allocation of FACTS devices in deregulated electricity market using genetic algorithms. In *Proceedings in 2004 IEEE Power Systems Conference and Exposition*, volume 1, pages 201–207, New York, U.S., Oct. 10-13 2004.

[138] E. J. Oliveira, J. W. M. Lima, and K. C. Almeida. Allocation of facts devices in hydrothermal system. *IEEE Transactions on Power Systems*, 15(1):276–282, Feb. 2000.

[139] R. Billinton and E Khan. A security based approach to composite power system reliability evaluation. *IEEE Transocean on Power Systems*, 7(1):65–72, 1992.

[140] V. Chankong and Y. Y. Haimes. *Multiobjective decision making theory and methodology*. North-Holland, New York, 1983.

[141] M. Reyes-Sierra and A. C. Coello. Multi-objective particle swarm optimisers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2:287–308, 2006.

[142] Q. H. Wu, Z. Lu, M. S. Li, and T. Y. Ji. Optimal placement of facts devices by a group search optimizer with multiple producer. *IEEE Congress on Evolutionary Computation*, pages 1033 – 1039, 2008.

[143] H. Akagi. New trends in active filters for power conditioning. *IEEE Transactions on Industry Application*, 32(6):1312–1322, November 1996.

[144] H. Ma and A. A. Girgis. Identification and tracking of harmonic sources in a power system using Kalman filter. *IEEE Transactions on Power Delivery*, 11:1659–1665, December 1996.

[145] Y. N. Fei, Z. Lu, W. H. Tang, and Q. H. Wu. Harmonic estimation using a global search optimiser. *Lecture Notes in Computer Science*, 4448:261–270, June 2007.

[146] M. Bettayeb and U. Qidwai. Recursive estimation of power system harmonics. *Electric Power Systems Research*, 47(2):143–152, 1998.

[147] Y. Taiyou and R. H. Lasseter. Stochastic optimal power flow: formulation and solution. *IEEE Power Engineering Society Summer Meeting*, 1:237–242, 2000.

[148] W. J. Tang, M. S. Li, Q. H. Wu, and J. R. Saunders. Bacterial foraging algorithm for optimal power flow in dynamic environments. *IEEE transactions on circuits and systems . I , Regular papers*, 55(8):2433–2442, 2008.