THE UNIVERSITY *of* LIVERPOOL

Optimisation Algorithms Inspired From Modelling Of Bacterial Foraging Patterns And Their Applications

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

W. J. TANG, B.Eng., M.Sc.(Eng.)

May 2008

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Q. H. Wu, for his continuous support and illuminating guidance. His passion for academic research has a deep influence on me. I have greatly benefitted from his multi-disciplinary knowledge, without which this research would surely have been very difficult to carry on. I am also grateful to Professor J. R. Saunders for his invaluable help and advice provided throughout my PhD study.

I want to thank members of Intelligence Engineering and Automation Group for all their help, support, interest and valuable suggestions. Especially I am obliged to Mr. M. S. Li and Dr A. Shintemirove, for their full support in our cooperation works. Dr S. He has provided me with the toolbox for the simulation of optimal power flow and Dr Z. Lu has offered his precious help and hints in the modelling work.

My thanks also go to the Department of Electrical Engineering and Electronics in the University of Liverpool, for providing the research facilities that make it possible for me to conduct this research. I am indebted to the University of Liverpool for the University scholarships (2004 – 2007), the Universities UK for the 'Overseas Research Students Awards Scheme' (2004 – 2007) and the British Federation of Women Graduates for the Main Grant (2006 – 2007).

Finally, special thanks to my friends, wherever they are, for supporting my study. In particular I am heartily indebted to my mum and dad, for their encouragement, patience and love, given over the period of my study.

Abstract

Research in biologically-inspired optimisation has been flourishing over the past decades. This approach adopts a bottom-up viewpoint to understand and mimic certain features of a biological system. It has been proved useful in developing nondeterministic algorithms, such as Evolutionary Algorithms (EAs) and Swarm Intelligence (SI). Bacteria, as the simplest creature in nature, are of particular interest in recent studies. In the past thousands of millions of years, bacteria have exhibited a self-organising behaviour to cope with the natural selection. For example, bacteria have developed a number of strategies to search for food sources with a very efficient manner. This thesis explores the potential of understanding of a biological system by modelling the underlying mechanisms of bacterial foraging patterns and investigates their applicability to engineering optimisation problems.

Modelling plays a significant role in understanding bacterial foraging behaviour. Mathematical expressions and experimental observations have been utilised to represent biological systems. However, difficulties arise from the lack of systematic analysis of the developed models and experimental data. Recently,. Systems Biology has been proposed to overcome this barrier, with the effort from a number of research fields, including Computer Science and Systems Engineering. At the same time, Individual-based Modelling (IbM) has emerged to assist the modelling of a biological system. Starting from a basic model of foraging and proliferation of bacteria, the development of an IbM of bacterial systems of this thesis focuses on a Varying Environment BActerial Model (VEBAM). Simulation results demonstrate that VEBAM is able to provide a new perspective to describe interactions between the bacteria and their

food environment.

Knowledge transfer from modelling of bacterial systems to solving optimisation problems also composes an important part of this study. Three Bacteria-inspired Algorithms (BaIAs) have been developed to bridge the gap between modelling and optimisation. These algorithms make use of the self-adaptability of individual bacteria in the group searching activities described in VEBAM, while incorporating a variety of additional features. In particular, the new bacterial foraging algorithm with varying population (BFAVP) takes bacterial metabolism into consideration. The group behaviour in Particle Swarm Optimiser (PSO) is adopted in Bacterial Swarming Algorithm (BSA) to enhance searching ability. To reduce computational time, another algorithm, a Paired-bacteria Optimiser (PBO) is designed specifically to further explore the capability of BaIAs. Simulation studies undertaken against a wide range of benchmark functions demonstrate a satisfying performance with a reasonable convergence speed. To explore the potential of bacterial searching ability in optimisation undertaken in a varying environment, a dynamic bacterial foraging algorithm (DBFA) is developed with the aim of solving optimisation in a time-varying environment. In this case, the balance between its convergence and exploration abilities is investigated, and a new scheme of reproduction is developed which is different from that used for static optimisation problems. The simulation studies have been undertaken and the results show that the DBFA can adapt to various environmental changes rapidly.

One of the challenging large-scale complex optimisation problems is optimal power flow (OPF) computation. BFAVP shows its advantage in solving this problem. A simulation study has been performed on an IEEE 30-bus system, and the results are compared with PSO algorithm and Fast Evolutionary Programming (FEP) algorithm, respectively. Furthermore, the OPF problem is extended for consideration in varying environments, on which DBFA has been evaluated. A simulation study has been undertaken on both the IEEE 30-bus system and the IEEE 118-bus system, in comparison with a number of existing algorithms. The dynamic OPF problem has been tackled for the first time in

the area of power systems, and the results obtained are encouraging, with a significant amount of energy could possibly being saved. Another application of BaIA in this thesis is concerned with estimating optimal parameters of a power transformer winding model using BSA. Compared with Genetic Algorithm (GA), BSA is able to obtain a more satisfying result in modelling the transformer winding, which could not be achieved using a theoretical transfer function model.

Contents

List of Figures

List of Tables

List Of Abbreviations

ACO	Ant Colony Optimisation
ACS	Ant Colony System
AS-TSP	Ant System-TSP
BacSim	SIMulator for IbM of BACterial colony growth
BaIA	Bacterial-inspired algorithm
BC	Bacterial Chemotaxis
BFA	Bacterial Foraging Algorithm
BFAVP	Bacterial Foraging Algorithm with Varying population
BIA	Biologically-inspired Algorithm
BSA	Bacterial Swarming Algorithm
CLPSO	Comprehensive Learning PSO
CMA-ES	Covariance-matrix Adaptation Evolution Strategy
COSMIC	Computing Systems of Microbial Interactions and Communications
CPSO	Constriction factor approach PSO
DBFA	Dynamic Bacterial Foraging Algorithm
DE	Differential Equation
EA	Evolutionary Algorithm
EC	Evolutionary Computation
E.coli	*Escherichia coli*
EN	Elestic Net algorithm
EP	Evolutionary Programming
ES	Evolution Strategy

FDR-PSO	Fitness-Distance-Ratio-based PSO
FEP	Fast Evolutionary Programming
FIPS	Fully informed Particle Swarm
FRA	Frequency Response Analysis
GA	Genetic Algorithm
GP	Genetic Programming
IbM	Individual-based Modellling
INDISIM	Individual Discrete Simulations
MPB	Moving Peaks Benchmark
NPGA	Niched Pareto Genetic Algorithm
OPF	Optimal Power Flow
PbM	Population-based Modelling
PBO	Paired-bacteria Optimiser
PSO	Particle Swarm Optimiser
PSOPC	Particle Swarm Optimiser with Passive Congregation
QM	Quantum Mechanical
QS	Quorum Sensing
RUBAM	RUle-based BActerial Modelling
SA	Simulated Annealing
SBML	Systems Biology Markup Language
SOM	Self-Organising Maps
SI	Swarm Intelligence
TSP	Travelling Salesman Problem
UPSO	Unified PSO
VEBAM	Varying Environment BActerial Model
VEGA	Vector Evaluated Genetic Algorithm

Nomenclature

f_{n}	nutrient distribution of bacterial foraging environment
β	boundary characteristic of bacterial foraging environment
C^l_{gain}	nutrient obtained by the l^{th} bacterium in grid (i, j) per step
p^t_l	position of the l^{th} bacterium at time t
ε^t_l	energy of the l^{th} bacterium at time t
φ^t_l	status of the l^{th} bacterium at time t
τ^t_l	the action to be adopted by the l^{th} bacterium at time t
λ^t_l	the probability of tumble of the l^{th} bacterium at time t
δ^t_l	run length of of the l^{th} bacterium at time t
ϕ^t_l	orientation angle of the l^{th} bacterium at time t
D_k	population density of the k^{th} cluster
G_k	position of the centre of gravity of the k^{th} cluster
F_k	diffusion exponent of the autoinducers in the k^{th} cluster
A_k	amount of autoinducers emitted by the k^{th} cluster with the mechanism of quorum sensing
a_{p}	amplitude of the peak of the autoinducer congregation
α	metabolism factor for quorum sensing
ϵ	neighbourhood radius of a cluster
v	velocity of each bacterium
γ	duration of the trajectory of each bacterium
$C(i)$	length of a unit walk of the i^{th} bacterium
J_i	fitness of the i^{th} bacterium
ϕ_{max}	maximal tumble angle of a bacterium

β	coefficient for energy transform of bacterium for metabolism
η	counter of bacterium's age
P_p	probability density of lifespan for a group of bacteria
N_c	number of chemotactic steps in a bacterium's lifetime circle
N_s	number of selection loops in bacterial chemotaxis
B	length vector of the boundaries of a search domain
g	pseudo gradient in a search domain
τ	frequency of changes in a dynamic environment
E_L	real-time tracking error
θ_{ij}	voltage angle difference between buses 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.)
N_0	set of numbers of total buses excluding slack bus
N_B	set of numbers of total buses
N_C	set of numbers of shunt compensators
N_E	set of numbers of network branches
N_G	set of numbers of generator buses
N_i	set of numbers of buses adjacent to bus i, including bus i
N_{PQ}	set of numbers of PQ buses
$N_Q^{\lim}$	set of numbers of buses on which injected reactive power outside limits
N_T	set of numbers of transformer branches
$N_V^{\lim}$	set of numbers of buses on which voltages outside limits
P_{D_i}	demanded active power at bus i (p.u.)
P_{G_i}	injected active power at bus i (p.u.)
Q_{C_i}	reactive power source installation at bus i (p.u.)
$N_E^{\lim}$	set of numbers of buses on which the apparent power is limited
Q_{D_i}	demanded reactive power at bus i (p.u.)
Q_{G_i}	injected reactive power at bus i (p.u.)
V_{G_i}	voltage vectors of PQ buses (p.u.)
T_i	tap position at transformer i

V_i	voltage magnitude at bus i (p.u.)
S_k	apparent power in branch k (p.u.)
C_s	series capacitances of the winding sections
C_g	ground capacitances of the winding sections
G_s	series conductances of the winding sections
G_g	ground conductances of the winding sections
L	inductances of the winding sections
R	resistances of the winding sections
M	mutual inductance between winding sections

Chapter 1

Introduction

1.1 Motivation

Over the last two decades, a large number of researchers have been working in the area of EA and SI. They have developed various optimisation algorithms stemming from the studies of genetic evolution, originally investigated by Fraser [1] and Friedberg [2]. These algorithms mainly offer stochastic search capabilities and carry out population-based calculation. They generally have a better performance for complex multi-modal optimisation problems in contrast with the conventional gradient-based techniques. Therefore they have been applied to solve various science and engineering problems. Currently there are still many researchers working in this area to improve the methodologies for specific optimisation issues. However, EA and SI algorithms are notorious for their computational load. Due to their time-consuming characteristic, their applicability has been hampered for large-scale systems, such as power systems, telecommunication networks, traffic systems and biological data processing. Many researchers have considered ways of improving the computational efficiency and also the capability for global searching and convergency of these algorithms. Currently there are two directions which could be taken to tackle these problems. One is to improve the algorithm from a mathematical point of view, which no doubt can make an optimisation algorithm perform better

for a specific problem. However, it does not require the understanding of the biological background from which the original methodology was derived. In this sense, although many EAs have been developed over the past decades, these algorithms share the same biological understanding which has not been deepened since the original GA and PSO arrived. We believe that further advances of these methodologies will require the knowledge of modelling biological systems and the knowledge transfer from the biological science, with support from mathematical analysis. The other way lies in the study of modelling of biological systems, which will play a significant role in the advance of knowledge in this field and provides great opportunities to develop novel optimisation techniques. This is why studying biological system modelling is necessary for creating new optimisation techniques.

Biology is traditionally an experiment-based subject and many biological problems are increasingly conceptual [3]. However, in recent years, novel techniques have been brought into this subject that allow us to study many variables in the biological system simultaneously, from different perspectives and with increasing accuracy. Generally, these techniques include qualitative or rule-based formalisms, which may provide sufficiently predictive models at 'higher levels', describing the emergent behaviour; and agent (individual)-based simulations that may be better suited to model the interactions of a large number of components. The emergence of these technologies triggered the study of an interdisciplinary subject, Systems Biology, as shown in Figure 1.1 [4]. There is a state-of-the-art approach in this study, called 'modelling the heart' [5]. It simulates from genes and cells to the whole organ, including the protein genome (*i.e.* individual proteins that genes code for), and the level of protein interaction within the context of subcellular, cellular, tissue, organ and system structures.

Following the success of this subject, many researchers have begun to investigate a demonstration system to describe phenomena observed across the life structure from the 'lower levels' to 'higher levels'. The life system of bacteria, as the simplest existing life species, is a reasonable starting point. There have

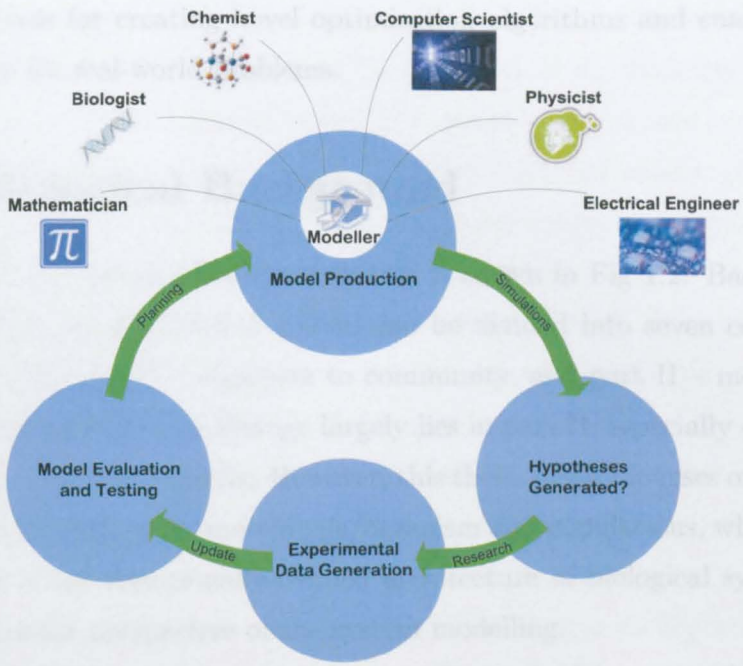


Figure 1.1: Systems Biology

been ordinary discrete equations describing those phenomena but with limited success; on the other hand, the data currently available do not permit the use of well-established systems engineering tools for parametric system identification. Nonetheless, with the emergence of computerised simulation of bacterial systems, this task has finally become feasible. Literature published in the past ten years has euphorically reported some successful models of bacterial foraging system [6]. Up until now, this work is still in the frontier of biological system modelling and requires even more inputs from various perspectives.

Based on theoretical studies reported in the literature and also previous work in individual-based modelling, we would like to further explore the research on modelling of bacterial foraging patterns and based on it, develop more practical and powerful biologically inspired algorithms which can be used for solving complex optimisation problems. We believe that this study will broaden the vision of researchers working in the areas of EAs and SI and it will

lay out a basis for creating novel optimisation algorithms and enabling their applications for real-world problems.

1.2 Biological Background

The architecture of a biological system is shown in Fig 1.2. Basically, the hierarchical levels of a normal animal can be divided into seven components with two parts: part I – organism to community, and part II – molecular to organ. Currently, Systems Biology largely lies in part II, especially concerning organelles/molecular and cells. However, this thesis mostly focuses on the three shaded components: cells, individuals/organism and populations, which breaks the barrier of the traditionally-defined architecture of biological system, and provides another perspective of the system modelling.

In a biological system, one of the questions raised by researchers is how we can obtain information for studying the interaction between system components. The answer appears straightforward. First of all, we have to define basic structures of the biological network in a living cell, and tackle the problem with the biological system responding to changing conditions and its robustness and stability. Then, we could predict the system behaviour based on the model of this biological system.

In recent years, there has been a rapid growth in biological databases. Models of cells, tissues, and organs, and the development of powerful computing hardware and algorithms have made it possible to explore functionality in a quantitative manner all the way from levels of genes to whole organs and systems. Although in many cases, the cellular, organ, and system functions of genes and proteins are unknown, clues sometimes come from homology in the gene sequences and other patterns being investigated by bioinformatics and its related fields. For example, at the individual model level, there are some well-known models in existence. Fleming *et al.*, developed a model called 'ATLSS' [7], which is an across trophic level system simulation. Railsback *et al.* investigated an individual-based Model for fish groups [8]. UIUC's Imaging

Systems Laboratory provided 'SmartForest', which deals with the management issues at the level of individual trees [9]. Schimitz *et al.* modelled ecosystem dynamics with 'Gecko', a well-known individual-based simulation tool [10]. 'Bacsim', investigated by Kreft *et al.*, models bacterial colony growth [11]. Furthermore, Reynolds proposed 'boids' to simulate coordinated group motion [12].

Generally speaking, there are two approaches in System Biology. The first approach is from a top-down point of view: how the activities and communication networks are implemented in living cells, and how they are ultimately encoded in genomes (*e.g.* vast perturbation experiments coupled to high-throughput functional experiments). The second approach is from both microscopic and macroscopic points of view: how proteins interact with each other at different levels and how different simulation approaches would be appropriate at different levels of description. For example, at the highest level, qualitative network analysis of biological pathways (*e.g.* with Petri nets) and quantitative network analysis (*e.g.* using the Monte Carlo approach). There are also two methods for carrying out molecular simulations of protein-protein interactions. The first is quantum mechanical (QM) predictions of chemical reaction energies, which is the most accurate method. The second is classical (ball-and-spring) force field to describe the atomic interactions [13]. At the moment, Systems Biology represents a 'molecular biology revolution', while there are still developments necessary to transform biology into a 'systems science'. Systems Biology will be extended to the higher levels in biological architecture, and will also require significant expertise and resources that cross traditional disciplinary boundaries. Also needed is the development of new theories and mathematics, as well as the development of new algorithms, their implementation on high-performance computer systems, and extensive use of large, distributed, and heterogeneous databases with wide availability to make software and computer systems usable. Ultimately, this will lead to a new model for biological analysis that will involve a cycle beginning with the computational synthesis of available biological information to formulate specific

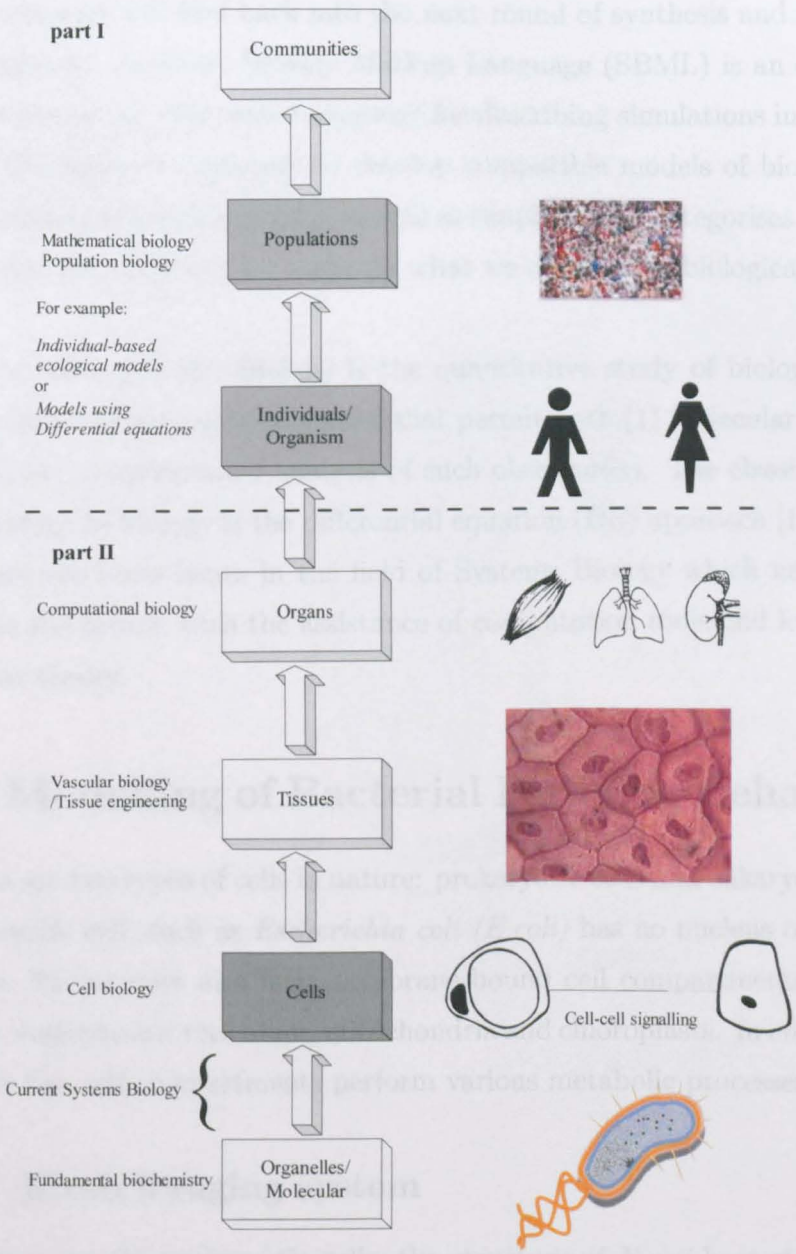


Figure 1.2: Architecture of biological system

biological hypotheses that will drive new experiments or, in some cases, specific computational simulations in place of experiments [14]. The data from the new experiments will feed back into the next round of synthesis and hypothesis development. Systems Biology Markup Language (SBML) is an early approach, which is an XML-based language for describing simulations in Systems Biology. It enables researchers to develop compatible models of biochemical network simulation/analysis tools, as well as simplifies and categorises the components and interactions that make up what we define as a 'biological system' [15].

In summary, Systems Biology is the quantitative study of biological systems, aided by technological advances that permit both (1) molecular observations and (2) computational analysis of such observation. The classical modelling strategy in biology is the differential equation (DE) approach [16]. However, there are some issues in the field of Systems Biology which need to be tackled in the future, with the assistance of computation tools and knowledge of systems theory.

1.3 Modelling of Bacterial Foraging Behaviour

There are two types of cells in nature: prokaryotic cells and eukaryotic cells. A prokaryotic cell, such as *Escherichia coli (E.coli)* has no nucleus or nuclear envelope. Prokaryotes also lack membrane-bound cell compartments such as vacuoles, endoplasmic reticulum, mitochondria and chloroplasts. In eukaryotes, the latter two cell compartments perform various metabolic processes.

1.3.1 *E.coli* foraging system

Bacteria are the prokaryotic cells, the structure of *E.coli* bacteria is illustrated in Fig. 1.3. It is shown in Fig. 1.3 that a prokaryotic cell has three envelopes. The Pilus connect the bacterium to another of its species, or to another bacterium of a different species, and build a bridge between the cytoplasm of both cells. Mesosomes may play a role in cell wall formation during

cell division and/or chromosome replication. The Nucleoid is an irregularly-shaped region within the cell where the genetic material is localised. The nucleic acid is a circular, double-stranded piece of Deoxyribonucleic acid (DNA), and multiple copies may exist. Ribosomes are complexes of Ribonucleic acid (RNA) and protein which catalyse the assembly of individual amino acids into polypeptide chains; this involves binding a messenger RNA and then using this as a template to join together the correct sequence of amino acids [17].

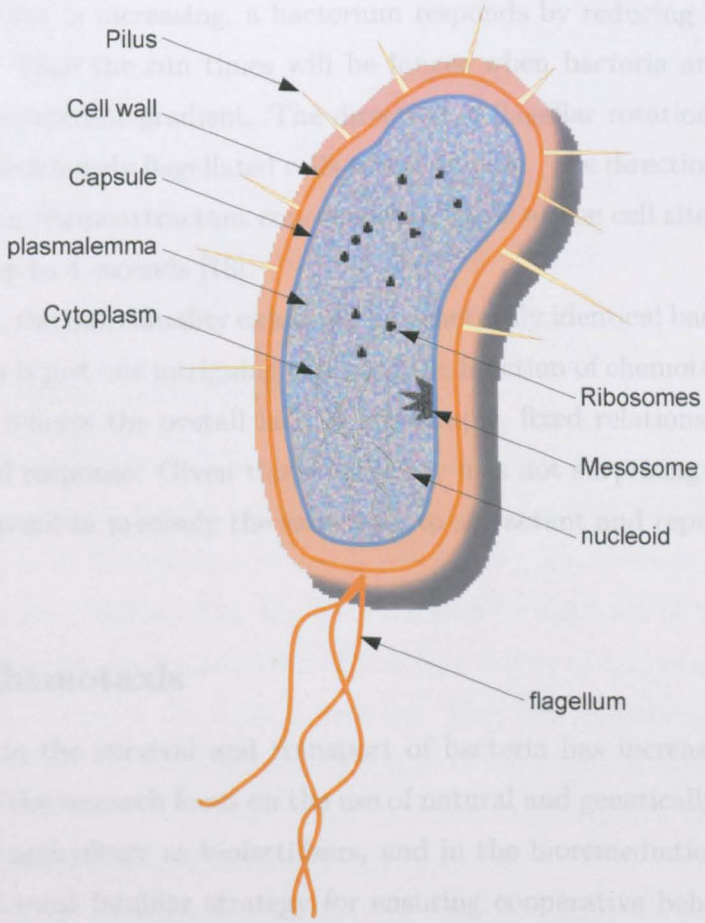


Figure 1.3: The structure of a prokaryotic cell

Bacteria swim by rotating thin, helical filaments known as flagella driven by a reversible motor embedded in the cell wall. Peritrichously flagellated bacteria such as *E.coli* have 8∼10 flagella placed randomly on the cell body.

The motion of individual peritrichously flagellated bacteria can be described in terms of run intervals during which the microbe swims approximately in a straight line interspersed with tumbles so that the organism undergoes a random reorientation. For *E.coli* the run and tumble durations are exponentially distributed with means of approximately 1 second and 0.1 second respectively in isotropic medium. Chemotactic bacteria sense temporal changes in concentration levels of chemical attractants and repellents. If the concentration level of an attractant is increasing, a bacterium responds by reducing its tumbling probability. Thus the run times will be longer when bacteria are swimming up a chemoattractant gradient. The direction of flagellar rotation determines whether peritrichously flagellated cells run or tumble. The direction of rotation depends upon chemoattractant concentration levels at the cell site over a time interval of up to 4 seconds [18].

However, the individuality exhibited by genetically identical bacteria within a population is just one intriguing aspect of the function of chemotaxis receptor arrays that reflects the overall lack of any simple, fixed relationship between stimulus and response. Given this complexity it is not surprising that no two bacteria respond in precisely the same way to attractant and repellent stimuli [19].

1.3.2 Chemotaxis

Interest in the survival and transport of bacteria has increased in recent years due to the research focus on the use of natural and genetically-engineered microbes in agriculture as biofertilisers, and in the bioremediation of organic toxins. The most familiar strategy for ensuring cooperative behaviour is attractive chemotaxis.

Swimming was found to consist of smooth 'runs' interrupted roughly every second by transient 'tumbles' (or 'twiddles'), which is illustrated in Fig. 1.4 [17]. Chemotaxis – the ability of the cells to move toward distant sources of food molecules – is based on the suppression of tumbles in cells that happened by chance to be moving up the gradient [20].

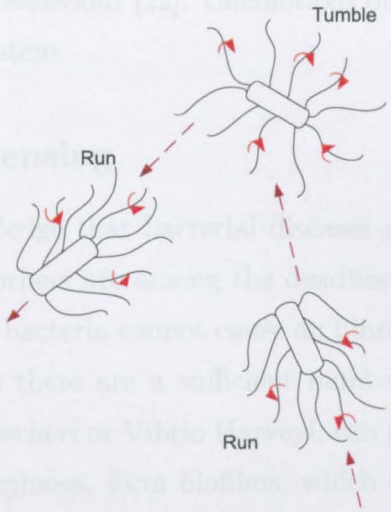


Figure 1.4: Bacterial foraging patterns

1.3.3 Sensory adaptation

As stated in Section 1.3.2, Chemotactic bacteria such as *E.coli* adapt to their environment by regulation of the direction of flagellar rotation. Adaptation can be simply defined as a change in the relationship between environmental stimulus and response that has been induced by the level of the stimulus. Sensory information is transduced in *E.coli* by a family of transmembrane protein that are located in the inner membrane. They are chemosensory receptor molecules that act via intermediates to inhibit the reversals of flagellar rotation that cause them to tumble. Thus, when the receptors are active, the swimming bacteria tumble and change direction less often [21].

As a result of adaptation, the bacteria are only inhibited from tumbling when responding to attractant gradients. Thus bacteria tend to move up concentration gradients, towards increasing concentrations of attractants. Conversely, tumbling frequency is increased to enhance the probability of moving away from an increasing concentration of repellent. Several minutes after responding to a stimulus, cells adapt to their environment by returning to their

prestimulus pattern of behaviour [22]. Chemotaxis of *E.coli* is one of the best-charaterised sensory system.

1.3.4 Quorum sensing

It is common knowledge that bacterial diseases such as cholera, anthrax, meningitis, and many others are among the deadliest in the world. However, it may be the case that bacteria cannot cause an illness when they are in small quantities. Only when there are a sufficient number of them can they act. Some, such as Vibrio Fischeri or Vibrio Harveyi, can glow in the dark. Others, like Pseudomonas Aeruginosa, form biofilms, which are composed of millions of microorganisms on the surface of human organs, and attack virulently those organs multiplying with tremendous speed, making it practically impossible for antibiotics to interfere [23].

Quorum sensing (QS) is widespread; it occurs in numerous Gram-negative and Gram-positive bacteria. In general, processes controlled by quorum sensing are ones that are unproductive when undertaken by an individual bacterium but become effective when undertaken by the group. For example, quorum sensing controls bioluminescence, secretion of virulence factors, sporulation, and conjugation. Thus, quorum sensing is a mechanism that allows bacteria to function as multi-cellular organisms.

In Miller and Bassler's discovery, quorum sensing is a process that allows bacteria to search for similar cells in their close surroundings using secreted chemical signaling molecules called autoinducers [23]. This is also called 'cell-cell communication'. Other bacteria release the same autoinducers in response. One-cell organisms in effect become multi-cellular organisms and can act together. This process enables a population of bacteria to collectively regulate gene expression and, therefore, behavior. In quorum sensing, bacteria assess their population density by detecting the concentration of a particular autoinducer, which is correlated with cell density. This 'census-talking' enables the group to express specific genes only at particular population densities.

1.3.5 Mutation

Bacteria are capable of surviving in a wide range of seemingly impossible situations. In the main part this is because very large numbers of individual cells are involved, so that even very rare mutations will occur often enough to solve the most acute problems and permit growth. But in addition, bacteria have evolved in an environment that has fluctuated so often in the past that they may have evolved a group of 'last ditch' mechanisms to meet these challenges. These include developing a metabolically inactive state; activating previously evolved, but silent genes; increasing rates of mutation under dire conditions; and favouring movement of exogenous and endogenous genetic elements. Together these survival mechanisms constitute the 'catastrophe insurance' of the cells.

1.4 Biologically-inspired Optimisation

1.4.1 Evolutionary algorithms

Genetic Algorithm

In the 20th century, Artificial Intelligence was one of the cutting-edge research fields. Over the second period of this century, many methodologies have been investigated to explore the similarity between natural evolution and problem-solving algorithms, one of which is GA. The beginning of the GA's development can be traced back to the early 1950s when computers were used for modelling and simulation studies of biological systems, such as the work of Bremermann [24]. However, the work undertaken in the late 1960s and the early 1970s at the University of Michigan, by Holland, first brought GA to the attention of a wider audience [25]. GA is a stochastic optimisation algorithm with a global search potential. The methodology is inspired by a biological metaphor and applies a pseudo-Darwinian process to evolve good solutions to real-world problems. As one of the major members of the EA family, GA adopts a populational unit of analysis, wherein each member of

the population is encoded as a potential solution to an optimisation problem. GA developed at the early stage used fixed-length binary strings and only two basic genetic operators: binary mutation and binary crossover, which are the major operators in 'schema' introduced in 1968 [26]. Evolution in the population of encodings is simulated by means of a pseudo-natural selection process using differential-fitness selection and pseudo-genetic operators which induce variation in the population in successive generations. In addition to the real-coded GAs implemented in the 1990s, the contribution to the GA theory itself also includes trying to create a generic framework for constrained optimisation problems [27] and combining the fuzzy logic, variable population size and reinitialisation strategy together with a simple GA to enhance the performance [28]. Optimisation of the mutation and crossover probabilities as control parameters of a GA was also investigated as a controlled Markov process [29]. Another branch of GAs lies in multi-objective optimisation problems. Vector Evaluated Genetic Algorithm (VEGA) was introduced by Schaffer [30]. Since then, a number of GA-based algorithms have been investigated to obtain the Pareto-optimal solution, such as the Niched Pareto Genetic Algorithm (NPGA) [31] and an extended multi-objective GA investigated by Rodriguez-Vazquez [32].

The applications of GAs include optimal reactive power dispatch [33], job-shop scheduling [34], sensor-based robot path planning and the training of radial basis function networks [35]. More recently, applications of GAs with low computational load in image processing have also been reported [36].

Evolutionary Programming

Modification in chromosome structures has to be taken to deal with non-trivial constraints which GAs have a certain difficulty to cope with [37]. To this end, both richer data structures and applicable genetic operators for these structures are needed. One example of using non-string chromosome representation and problem specific genetic operators is Evolutionary Programming (EP) developed by Fogel [38].

By incorporating problem-specific knowledge in the chromosome structure,

EP can accommodate a wide range of evolution-based systems. Similar to GA, it also maintains a population of individuals, and each individual represents a potential solution to the problem. Each solution is evaluated to give a measure of its 'fitness'. Then, a new population is formed by selecting the fitter individuals, based on the ranking of these individuals. In contrast to GAs, EP does not conduct a crossover operation, but it fully capitalises on the mutation operation in the evolution process. For each individual, mutation varies in severity, which affects the behaviour of the individual. Thus, EP focuses on optimising continuous functions with only mutation involved, by solely modelling the behavioural linkage between parents and their offsprings, which are directly coded as numerical numbers. After a certain number of generations, the EP converges - it is hoped that the best individual represents a near-optimum or reasonable solution [37]. The contemporary variants of EP lie in the use of Cauchy mutation instead of Gaussian mutation for a broader range of variation, which greatly enhances the global search capability of EP [39]. EP has been adopted to optimise parameters of an on-chip voltage reference circuit [40], and it has also been successfully applied to reactive power dispatch [41].

Evolution Strategy

Evolution strategy (ES) was initially invented in the 1960s for the purpose of parameter optimisation by Rechenberg [42] and Schwefel [43]. Schwefel was the first researcher to substitute a discrete mutation mechanism with normally distributed mutations. The initially two membered ES works by creating one real-valued vector of object variables from its parent by applying mutation with an identical standard deviation to each object variable. Then, the resulting individual is evaluated and compared to its parent, and the better one survives to become a parent of the next generation, while the other is discarded. This simple selection mechanism is fully characterized by the term $(1+1)$-ES. However, due to the lack of the variation brought by a multimembered population, Recherburg added the notion of population to $(1 + 1)$-ES and introduced a multi-member ES with parents, denoted as $(\mu + 1)$-ES [44]. In $(\mu + 1)$-ES, μ

parent individuals are recombined to form one offspring, which after being mutated eventually replaces the worst parent individual, if it is better. However, this strategy has never been widely used.

There are other variants of this type of ES, such as $(1 + \lambda)$-ES, where λ mutants can be generated and compete with the parent, and $(\mu, ?)$-ES with all parents selected in every generation with a varying population, $(\mu+\lambda)$-ES with the best μ individuals out of the union of parents and offspring surviving, and (μ, λ)-ES with only the best μ offspring individuals forming the next parent generation. Rudolph also modelled the mutation of ES using a Markov Chain [45]. Currently, the (μ, λ)-ES characterises state-of-the-art in ES research.

ES has been applied to parameter estimation [46], image processing [47] and computer vision system [48]. Task scheduling and car automation have also been tackled by ES [49].

Genetic Programming

Another paradigm of EAs is Genetic Programming (GP). GP was initially developed to solve various complex optimisation and search problems by Koza [50]. Many seemingly different problems in artificial intelligence, symbolic processing and machine learning can be viewed as requiring discovery of a computer program or functional structure that produces desired outputs for particular inputs. Solving these problems can be reformulated as a search for a highly fitted individual of computer program or functional structure in the feasible search domain.

GP traditionally distinguishes itself from GA in two fundamental ways. Instead of evolving binary strings which represent an indirect encoding of a potential solution, a search is applied to the solution directly in GP, and a solution in this case could be a computer program. The second fundamental difference is in the variable-length representation adopted by GP. With GA we normally adopt a fixed-length encoding, whereby the number of genes is fixed that will comprise an individual at the outset of a run. In GP, it is recognised that the length of a solution program may not be known as a priori, thus,

the number of genes must itself be open to evolution. Initialisation of a GP population consequently attempts to generate diversity not only in the values of the genes (the primitive symbols of the programming language) but also in the structure of the individuals [51]. GP has been adopted for a number of applications. The first application which appeared in the literature is the automated synthesis of analog electrical circuits using GP proposed by Koza [52]. There are other fields reporting promising results obtained by GP, such as breast cancer diagnosis [53].

1.4.2 Swarm Intelligence

SI has been developed alongside EAs. Two of the most well-known strategies in this area are PSO, Ant Colony Optimisation (ACO). These trajectory tracking algorithms are inspired from the collective behaviour of animals, which exhibit decentralised, self-organised patterns in the foraging process.

Particle Swarm Optimiser

PSO is a population-based stochastic optimisation technique developed by Eberhart and Kennedy in 1995 [54], inspired by the social behaviour of bird flocking or fish schooling. Consider the following scenario: a group of birds are randomly searching for food in an area where there is only one piece of food. None of the birds knows where the food is, however, they know the position of the leader bird, *i.e.* the bird nearest to the food. Obviously, one of the effective strategies to find the food is to follow the leader bird.

PSO is based on this scenario to solve optimisation problems. In PSO, each single solution is a 'bird' in the search space or a 'particle' in the algorithm. All of the particles have their own fitness value which is evaluated by the fitness function to be optimised, and have velocities which direct the flying of the particles. The particles fly in the search space by following the current optimum particles.

The population of PSO is called a *swarm* and is initialised with certain positions and velocities. In each iteration, the position of an individual is up-

dated according to the historical best position of that individual $pbest_i$ and the global best position of the population in that iteration *gbest*. The position and the velocity are updated to improve the fitness function at each time step. When a particle discovers a pattern that is better than any previously found, the positional coordinates are stored in the vector $pbest_i$, the best position found by particle i so far. The difference between $pbest_i$ and the current position is stochastically appended to the current velocity. This causes a change to the trajectory the particle would take at that position. The stochastically weighted difference between the population's best position *gbest* and the individual's current position is also added to the velocity, in order to adjust for the step length of the next iteration. These adjustments direct the search around two best positions.

Another important variant of the standard PSO is the constriction factor approach PSO (CPSO) which was proposed by Clerc [55]. CPSO ensures the convergence of the dynamical system by making sure that the eigenvalues are no more than 1, whether from the algebraic point of view or the analytic point of view. The CPSO ensures the convergence of the search procedures and can generate higher quality solutions than the standard PSO using a modified inertia weight. Other variants of PSO have also been developed in recent years, such as Particle Swarm Optimiser with Passive Congregation (PSOPC) [56], Unified PSO (UPSO), fitness-distance-ratio-based PSO (FDR-PSO), Fully Informed Particle Swarm (FIPS) and most recently, Comprehensive Learning PSO (CLPSO) [57]. These algorithms have produced encouraging results in both benchmark testing and applications in the real world. There are many similarities between PSO and EAs. Both of them initialise solutions and update generations, however PSO has no evolution operators. In PSO, particles try to reach the optimum by following the current global optimum instead of using evolutionary operators, such as mutation and crossover.

In the past several years, PSO has been successfully applied in many research and application areas, such as multi-modal biomedical image registration [58] and the Iterated Prisoner's Dilemma [59]. It is claimed that PSO has

not only obtained satisfied results for continuous functions, but also demonstrated its stability in multidimensional complex spaces [55].

Ant Colony Optimiser

Another interesting approach in the area of optimisation was inspired by ant colony behaviour. During the past ten years, 'the social insect metaphor for solving problems has become a hot topic' [60]. By modelling and simulating ant foraging behaviour, brood sorting, nest building and self-assembling, *etc*, we would be able to develop algorithms which could be used for complex, combinatorial optimisation problems. There are several computational models inspired by the collective foraging behaviour of ants. They are named the ACO. Many ant species have trail-laying trail-following behaviour when foraging: individual ants deposit a chemical substance called pheromone as they move from a food source to their nest, and foragers follow such pheromone trails. The process, whereby an ant is influenced to move toward a food source by another ant or by a chemical trail, is called recruitment [60], and recruitment based solely on chemical trails is called mass recruitment. This pattern has been used to develop optimisation algorithms trying to solve the Travelling Salesman Problem (TSP), such as Ant System-TSP (AS-TSP), which attempts to solve TSP by a simulated ant system. The solution obtained by AS-TSP is much better than the one found by GA [60]. Another approach, called Ant Colony System (ACS) has been introduced by Dorigo and Gambardella to improve the performance of ant system [61]. It is based on four modifications of ant system: a different transition rule, a different pheromone trail update rule, the use of local updates of pheromone trail to favor exploration, and the use of a candidate list to restrict the choice of the next city to visit. Thus, this algorithm was introduced initially for solving TSP problems by the synergistic use of cooperation among many ants which communicate by distributed memory implemented as pheromone deposited on edges of a graph [61]. ACS was tested on problems of various sizes and compared with other algorithms, such as the elastic net algorithm (EN), self-organising maps (SOM), Simulated Annealing

(SA), GA and EP [61], for solving a TSP problem. The results, obtained by ACS for the problem with randomly generated sets of a symmetric 50 cities, were compared with those obtained by SA, EN and SOM respectively. ACS is able to achieve the minumum results in most cases in 2500 iterations with 10 ants. Furthermore, Dorigo and Gambardella introduced a local search procedure to be performed in combination with ACS in order to achieve better performance in larger problems [62].

ACO has a certain relationship with conventional optimisation algorithms. Cross-fertilisation of ACO and stochastic gradient descent was analysed in [63]. It also proves the convergence of ACO. To understand the mechanism of ACO more fundamentally, Theraulaz [64] proposed Stigmergy to explain the relationship between ACO and natural self-organising. Stigmergy is a method of indirect communication in a self-organising emergent system in which the individual parts communicate with one another by modifying their local environment. It explains how individuals work as if they are alone while their collective activities appear to be coordinated in an insect society. Nevertheless, all the optimisation problems dealt with in the previous sections are static: the problem to be solved does not vary over time. One desirable feature of the swarm-based approach is that it may allow for enhanced efficiency when the representation of the problem under investigation is spatially distributed and changing over time. Although it still needs to improve, the research of this kind of SI has a promising future with more applications in telecommunication networks [60] and other areas. The applications of ACO include TSP [62], vehicle routing [65], course timetabling [66], graph colouring [67] and protein folding [68]. The merits of ACO are well illustrated by these applications.

1.5 Thesis Outline

This thesis is structured as follows:

Chapter 2 presents a Varying Environment BActerial foraging Model (VEBAM), which focuses on the use of an Individual-based Modeling (IbM)

method to simulate the activities of bacteria at levels of cells and population. Under this architecture, the interactions between the environment and bacteria are investigated at both the levels of cells and population. A new algorithm describing bacterial chemotaxis is derived from the framework and simulation studies are undertaken to evaluate this algorithm. At the level of population, the mechanism of quorum sensing is studied, in order to obtain a deeper understanding of how and when this mechanism works and its influence on bacterial evolution. This chapter also presents the work of using VEBAM to simulate the 'cell-to-cell communication' incorporated in bacterial foraging behaviors, in both intracellular and population scales. The simulation results show that the proposed model can reflect the bacterial foraging behaviour and population evolution in varying environments, to a biologically-plausible degree.

Chapter 3 introduces the major challenges in bridging the gap between modelling and algorithms and illustrates the methodology of knowledge transfer between them. It is followed by the demonstration of three BaIAs: BFAVP with a mechanism for individual growth simplified from the VEBAM, BSA inspired from the foraging behaviour of *E.coli* bacteria, especially the population level and PBO, which is similar to PSO, but with the modified usage of computation in individual dimensions. The benchmark testing results of the above three algorithms are listed in each part respectively.

Chapter 4 presents another BaIA with a different goal, to demonstrate an optimisation methodology developed specifically for dynamic environment. Optimisation in dynamic environments has received great attention in recent years. In this chapter, an algorithm aiming at optimisation in dynamic environments, called DBFA, is investigated. A test bed proposed in the literature is adopted to evaluate the performance of DBFA. A range of random changes, which occur with different probabilities, are considered in a dynamic environment. The simulation results show that

DBFA can adapt to various environmental changes and provide satisfactory performance in terms of both accuracy and stability, in comparison with the conventional Bacterial Foraging Algorithm (BFA).

Chapter 5 describes three applications for which the BaIAs have been applied. The first two are concerning with OPF problem, in which optimisation is considered in both static and dynamic cases; the third is the power transformer winding problem. The simulation results show that these problems can be tackled successfully using the algorithms presented in the above chapters.

Chapter 6 concludes the thesis based on the outcomes obtained in this study, followed by a discussion of the challenges of this work. Ideas for future work are also listed in this chapter.

1.6 Contributions of The Research

The major contributions made in this research are highlighted in this section and the original investigations have been undertaken on the following aspects.

1. *An IbM approach for modelling of bacterial foraging patterns in a varying environment*: A novel IbM approach has been proposed for modelling of bacterial foraging patterns. An architecture, together with a mathematical framework, has been developed. In this architecture, the system components of chemotaxis, metabolism, proliferation, and quorum sensing have been designed associated with mathematical equations derived to describe these specific biological phenomena. Simulation work of VE-BAM has been undertaken. Comprehensive simulation results have been obtained which show that the model outputs have a great similarity in comparison with the real *E.coli* foraging patterns as described in the literature.

2. *A bacterial foraging algorithm with varying population*: This work has been undertaken in cooperation with fellow colleagues. More biological

details have been taken into consideration in developing BFAVP, in order to further reflect the behaviour of bacterial foraging patterns. By contrast to conventional population-based EAs, BFAVP has a varying population scheme. The algorithm is based on the understanding of bacterial proliferation and it contains the following components: chemotaxis, metabolism, quorum sensing and proliferation. The algorithm is not only biologically realistic but also more efficient in global search performance, especially in solving multi-modal problems, which has been observed by assessing the function evaluation undertaken on a number of benchmarks. The merits of the BFAVP are illustrated in comparison with other EAs.

3. *A bacterial swarming algorithm*: Based on the fundamental study of the aforementioned aspect, a novel optimisation algorithm called BSA has been developed. This algorithm incorporates the merits of swarm performance, with a simplified QS approach, to enhance the global search and convergency capabilities. It also keeps the major character of BFA. BSA has been evaluated on a number of benchmark problems, which include uni-modal and multi-modal functions in low and high dimension domains, respectively. The simulation results have shown that BSA has superior performance in comparison with FEP and PSO.

4. *A paired-bacteria optimiser*: In order to reduce the computation load of the bacterial foraging algorithm, PBO has been investigated. This algorithm builds on the merits of PSO, while using 'divide-and-conquer' strategy in dealing with high dimensional problems, but only employs two bacteria in a population for global search. Therefore, the computation load is significantly reduced, while keeping the satisfactory performance in solving global optimisation problems. Simulation studies have been undertaken to show the merits of the PBO in comparison with other EAs.

5. *Optimisation in varying environments*: A new concept of global optimisation in varying environments has been introduced, which is significantly

different from the conventional study of optimisation of static systems. The simulation study of a varying environment and in this environment, solving an optimisation problem propose a great challenge. In order to face this challenge, DBFA has been developed. By contrast to most of EAs which are developed for static optimisation problems, DBFA adopts not only the merits of local search which is inspired from the chemotactic process but also a new selection scheme which enables bacteria to flexibly adapt themselves to variation of the environment. The diversity is maintained in a certain level throughout the optimisation process. Two criteria have been introduced to evaluate the tracking performances of DBFA. It has been evaluated on a 'moving peaks benchmark' platform, and its satisfying performance has been observed.

6. *Optimal power flow in static form*: OPF problem is investigated in order to obtain optimised operating conditions within specific constraints. Thus, it can be formulated as a constrained large-scale high-dimensional optimisation problem. It has already been attempted by conventional gradient-based methods and more recently, non-conventional ones, such as EAs. BFAVP has been adopted to tackle this problem and the evaluation has been undertaken on IEEE 30-bus test system. The simulation results have demonstrated that BFAVP has a superior performance than PSO and FEP.

7. *Optimal power flow with load variations*: This is the first time to introduce a concept of solving an OPF problem with consideration of load changes alongside the optimisation process. A dynamic environment of a power system, which suffers from load changes randomly occurring on a number of buses simultaneously with a given probability, has been simulated. DBFA has been evaluated on two test systems: IEEE 30-bus test system and IEEE 118-bus system respectively. In order to evaluate the convergence rate and tracking ability of DBFA, two error tracking criteria have been designed: one is for evaluating the global tracking ability

of DBFA over the whole process of load variations, and the other focuses on the assessment of the convergence rate by counting the total number of function evaluations of DBFA. The simulation results, obtained by evaluating DBFA in comparison with BFA and PSO under the various scenarios of power system operation, have been presented and discussed. The results have demonstrated great potential of this study which may lead to a significant knowledge advance in this field in which the static OPF problems have been considered for more than half a century.

8. *Parameter optimisation of power transformer winding model*: This work was undertaken in cooperation with fellow colleagues. BSA has been applied to identify the parameters of a lumped parameter model of transformer winding. It includes search space estimation for the model parameters using analytical calculations and performing intelligent learning for determining the model parameters with BSA. This work shows that transformer winding models can be successfully constructed using BSA.

1.7 Autobibliography

List of the publications produced from this work:

1. **Tang, W. J.**, Wu, Q. H., Saunders, J. R., 'Bacterial Foraging Algorithm For Optimal Power Flow in Dynamic Environments', *IEEE Transactions on Circuits and Systems - Part I: Regular Papers.* Accepted.

2. **Tang, W. J.**, Wu, Q. H., 'Evolutionary Computation in Artificial Intelligence: A Review', *Transactions of the Institute of Measurement and Control.* Accepted.

3. Wu, Q. H., **Tang, W. J.**, He, S., Cao, Y. J., 'Dispatching', *Encyclopedia of Electrical Engineering and Electronics.* J. Webster (ed.), 2007. John Wiley & Sons, Inc.

4. **Tang, W. J.**, Wu, Q. H., Saunders, J. R., 'A novel model for bacterial foraging in varying environments', *2006 International Conference on Computational Science and its Applications (ICCSA 2006)*, Glasgow. May 2006. Lecture Notes in Computer Science, 3980: 556-565, Springer.

5. **Tang, W. J.**, Wu, Q. H., Saunders, J. R., 'Bacterial Foraging Algorithm For Dynamic Environments', *in Proceeding of 2006 IEEE Congress on Evolutionary Computation (CEC 2006).* Sheraton Vancouver Wall Centre, Vancouver, BC, Canada. July 16-21, 2006.

6. **Tang, W. J.**, Li, M. S., He, S., Wu, Q. H., Saunders, J. R., 'Optimal Power Flow With Dynamic Loads Using Bacterial Foraging Algorithm', *in Proceeding of 2006 International Conference on Power System Technology.* Chongqing, China, October 22-26, 2006.

7. **Tang, W. J.**, Wu, Q. H., Saunders, J. R., 'Individual-based Modeling of bacterial Foraging With Quorum Sensing in A Time-Varying Environment', *5Th European Conference On Evolutionary Computation Machine Learning And Data Mining In Bioinformatics, (EvoBio 2007)*, Valencia, Spain, April 2007. Lecture Notes in Computer Science, 4447: 280-290, Springer.

8. Li, M. S., **Tang, W. J.**, Wu, Q. H., Saunders, J. R., 'Bacterial Foraging Algorithm with Varying Population for Optimal Power Flow', *Applications of Evolutinary Computing, EvoWorkshops 2007: EvoCoMnet, (EvoCoMnet 2007)*, Valencia, Spain, April 2007. Lecture Notes in Computer Science, 4448: 32-41, Springer. (**Best Paper Award**)

9. **Tang, W. J.**, Wu, Q. H., Saunders, J. R., 'A Bacterial Swarming Algorithm For Global Optimization', *in Proceeding of 2007 IEEE Congress on Evolutionary Computation (CEC 2007).* Singapore. September 25-28, 2007.

10. Shintemirov, A., **Tang, W. J.**, Tang, W. H., Wu, Q. H., 'Improved Modeling of Power Transformer Winding Using Bacterial Swarming Al-

gorithm and Frequency Response', 9 pages. Submitted to *Electric Power Systems Research.*

11. **Tang, W. J.**, Wu, Q. H., Saunders, J. R., 'Individual-based Modeling of Bacterial Foraging Behaviors in Varying Environments', 25 pages. Ready to be resubmitted to *BioSystems.*

12. **Tang, W. J.**, Li, M. S., Wu, Q. H., Saunders, J. R., 'A Paired-bacteria Optimiser'. In preparation.

Chapter 2

Modelling of Bacterial Foraging Behaviour

2.1 Introduction

This chapter presents an architecture incorporating a mathematical framework for developing VEBAM. Our study focuses on the use of IbM to simulate the activities of bacteria at the levels of cells and populations. In particular, the interactions between the environment and bacteria are investigated at the level of cells. A new model describing bacterial chemotaxis is derived from the framework and simulation studies are undertaken to evaluate thismodel. At the population level, the mechanism of QS is studied, in order to obtain a deeper understanding of how and when this mechanism works and its influence on bacterial evolution. This study focuses on the development of VEBAM to simulate this 'cell-cell communication' incorporated in bacterial foraging behaviour, at both intracellular and population scales. The simulation results show that the proposed algorithm can reflect the bacterial foraging behaviors and population evolution in varying environments, to a biologically-realistic degree.

2.2 Individual-based Modelling

Interest in biologically-inspired modelling, directed at understanding the survival and reproduction of bacteria, has been stimulated by Systems Biology as an emerging discipline. Research in this area not only benefits the study of biological systems, but also aids the development of novel methodologies for solving modelling and optimisation problems for non-biological systems [69] [70]. One of the most important issues in biologically-inspired modelling is to understand the underlying mechanism of a biological system and to develop a generic model for bacterial behaviors.

IbM is one of the emerging approaches in Systems Biology. The basic idea of IbM is to simulate the behaviour or dynamics of an individual that interacts with others synchronously or asynchronously. IbM is commonly based on cellular automaton models, explaining qualitatively foraging pattern formations for different nutritional regimes [71]. The essential capability of an IbM enables the description of all the states, inputs and outputs of an individual and its relationship with other individuals living in the same population. In contrast to population-based modelling (PbM), IbM possesses a more flexible and robust capability for simulating a complex system where there are a large number of individuals that have their own behaviour or dynamics influenced by other individuals and the environment. One example is a SIMulator for IbM of BACterial colony growth (BacSim), a simulation tool based on understanding colonial growth in *E.coli*. It constructs a spatially explicit IbM of bacterial population growth that is quantitatively correct [11]. More recently, RUle-based BActerial Modelling (RUBAM) [72], which is an evolved version of IbM, introduces a fuzzy logic classifier system to characterise rules for bacterial movement. RUBAM utilises a rule-based architecture to investigate ecological and evolvable outcomes among very large numbers of artificial agents. The symbolic architecture of RUBAM employs a modified learning classifier system, which has been applied previously to biological systems. Agents in RUBAM that represent bacteria are constrained by rules that enact one or more phenotypes. They are able to adapt to changes in their environment through

implementing rule sets that mimic regulation of gene expression. In addition, agent populations can evolve by mutation and selection. The computational architecture employed also allows tracking of the life history and genealogy of individual agents within the environment [72].

Furthermore, there are other models, such as Individual Discrete Simulations (INDISIM) [73] and Computing Systems of Microbial Interactions and Communications (COSMIC) [74] [75], which are concerned with the details of cellular genetic interactions. COSMIC is an IbM approach to microbial ecology and evolution developed at The University of Liverpool. It seeks to encapsulate the behaviour of computational objects and processes with both biologically-plausible agent architectures and computational tractability. The computational architectures employed in these studies embrace agent-based models that employ symbolic (rule-based) systems such as classifier systems and sub-symbolic network-based architectures. This approach provides models for understanding adaptation and evolution in the communities of self-replicating agents representing bacteria living in virtual computational ecosystems [74].

The IbM can perform flexibly to simulate more complex ecological problems, considering the most important features of bacterial evolution. Moreover, the IbM does not require pre-understanding of the aggregate behaviour, which makes the model more expandable for various scenarios of biological systems.

However, bacterial foraging modelling involves high-throughput data processing work and poses a dilemma between biological reality and computational efficiency when modelling the details of a bacterial system. As a result, the models either represent these phenomena in a microscopic manner (*e.g.* BacSim and COSMIC only simulate metabolism and evolve bacterial growth) or apply an artificial mechanism or metaphor instead of biologically-plausible rules. Similar to these models, RUBAM employs fuzzy logic instead of chemotactic descriptions to represent foraging patterns. In this sense, a model that is able to represent a biological system with more knowledge of chemotactic processes, from a macroscopic point of view, is needed. In this chapter, we

propose VEBAM to simulate bacterial foraging patterns in a varying environment. Two important issues, bacterial foraging patterns at the cell level and quorum sensing at the population level, are addressed.

2.3 Chemotaxis

The foraging patterns of bacteria have been thoroughly investigated over the past few decades [6]. Chemotaxis is one of the most well studied strategies for ensuring cooperative behaviors in bacteria [20]. This strategy describes the ability of the cells to move toward distant sources of food molecules, by sensing temporal changes in concentration levels of chemical attractants and repellents. Swimming is found to consist of smooth 'runs' interrupted roughly every second by transient 'tumbles'. Chemotaxis is based on the suppression of tumbles in cells that takes place in the expectation of moving up a gradient of attractants. The study of this bacterial foraging pattern has proved fruitful for developing mathematical models [18] [21] [76] [77] [78]. These models describe and analyse bacterial phenomena from different viewpoints. For example, a mathematical and computational model at the microscopic level of bacterial motility and chemotaxis can be found in the work of Dillon *et al* [18]. The optimal chemotactic strategy of *E.coli* is analysed by evaluating the signal to noise ratio [21]. The bacterial chemotaxis signalling pathway is modelled and a full set of conditions are derived for this signalling system to achieve perfect adaptation to the environment [78].

In a bacterium such as *E.coli*, the chemotaxis pathway is well characterised, as shown in Fig. 2.1 [79]. Receptors at the cell surface detect changes in the concentrations of attractants and generate shifts in the level of phosphorylation of a diffusible signaling protein CheY. Phospho-CheY diffuses from CheA freely through the cell, and when it encounters a flagellar motor it binds to a flagellar protein called FilM.

Phosphorylated CheY modulates the direction of flagellar motor rotation and thus affects swimming behavior of the cell. Phospho-CheY bound to FilM

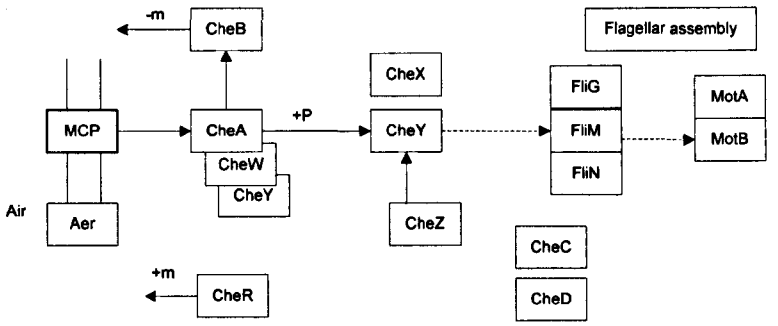


Figure 2.1: Chemotaxis pathway

induces tumbling by causing a change in the sense of flagellar rotation from counterclockwise to clockwise, as viewed from behind. The entire signal transduction pathway includes five attractant-specific receptors (Tsr, Tar, Trg, Tap, and Aer), six cytoplasmic chemotaxis proteins (CheA, CheW, CheR, CheB, CheY, and CheZ), and three proteins comprising a switch complex at the cytoplasmic face of the flagellar motor (FliM, FliN, and FliG). Based on genetic and biochemical data, functions have been assigned to all the chemotaxis proteins: CheA is a kinase that phosphorylates the response regulator CheY and also the methylesterase CheB; CheW is an adaptor protein, coupling CheA to receptors; CheR, a methyltransferase, and CheB, a methylesterase, mediate adaptation to a constant attractant concentration by adjusting the methylation level of receptors; CheZ is a phosphatase of CheY to CheP. Despite its relative simplicity, the chemotaxis system exhibits features common to many cellular networks, such as signal amplification, integration, and adaptation [80]. This has inspired a number of recent attempts to mathematically model this pathway.

The methylation state of the MCPs can thereby provide a memory mechanism that allows a cell to compare its present situation to its recent past. In that case, bacteria could detect a change in occupancy of the aspartate receptor as little as 0.1-0.2% [20]. As a result, chemotactic cells tumble less frequently if cells experience an increase in chemoattractant concentration over a period

of time. And it is known that for the chemoattractant aspartate, a cell compares the occupancy of a given receptor over the past second with that of the previous 3 seconds.

The tendency to tumble is enhanced when the bacterium perceives conditions to be worsening – when attractant concentrations decrease or repellent concentrations increase. Conversely, tumbling is suppressed and cells keep running when they detect that conditions are improving. Thus, when a bacterium runs up a gradient of attractants or down a gradient of repellents it tends to continue on course [71].

In general it is apparent that the change in stimulus concentration that a bacterium can detect is a constant fraction of the background stimulus intensity. This relationship, known as Weber's Law of psychophysics, is a general feature of animal sensory systems. It is interesting that it seems to apply as well to bacteria.

Recently, bacterial chemotaxis has become a honey pot for modellers. Numerous computer programs have been written to represent kinetic and other features of the pathway, with significant success. It also provides the fundamental knowledge on which this study is based.

2.4 Quorum Sensing

Living organisms are always subject to time-varying conditions in the form of environmental changes. That is why they develop their own essential ability to sense signals in the environment and adapt their behaviour accordingly. To understand the intracellular molecular machinery that is responsible for the complex collective behaviour of multicellular populations is an exigent problem of modern biology. A new branch of microbiology, quorum sensing, was discovered by Miller *et al.* (which is also called 'cell-cell communication') [23].

Cells that have their specific gene expression programs activated can produce autoinducers, *i.e.* signalling molecules, that initiate quorum sensing [81]. An 'on-off' gene expression switch controls this phenomenon in response to the

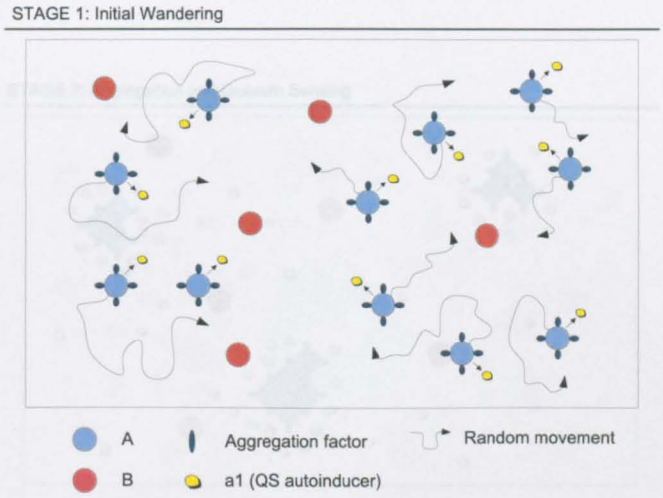


Figure 2.2: Quorum sensing (Step I)

increase of autoinducer concentration. Thus, once a certain level of a cluster density is reached, a quorum sensing network of the population is formed and turns to the 'on' state. It also amplifies the autoinducer signal of the 'quorum' cells. This signal then turns on the expression of the phenotype-specific genes of other cells and boots the production of autoinducers in the entire population. Once it turns to the 'on' state, it remains in this state until the density of the cluster falls below a certain critical level. The quorum sensing network remains in the 'off' state, until a certain concentration of the critical autoinducers is reached once more. Figures 2.2~2.6 illustrate the quorum sensing process within and between two engineered strains of *E.coli* [82].

Communication between bacteria, by the mechanism referred to as 'quorum sensing', is widespread in nature. Quorum sensing also controls bacterial behavior, especially those behaviour that would be unproductive when undertaken by an individual bacterium, but become effective in a cooperative group. Such phenomena as bioluminescence, secretion of virulence factors, sporulation and conjugation may be governed by quorum sensing. In this sense, bacteria are able to function as multi-cellular organisms. A general mathematical model of quorum sensing is introduced in bacteria [83]. A mathematical model of

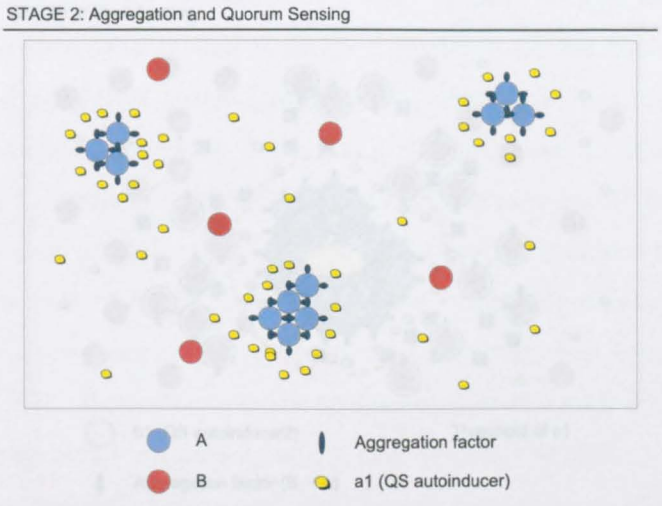


Figure 2.3: Quorum sensing (Step II)

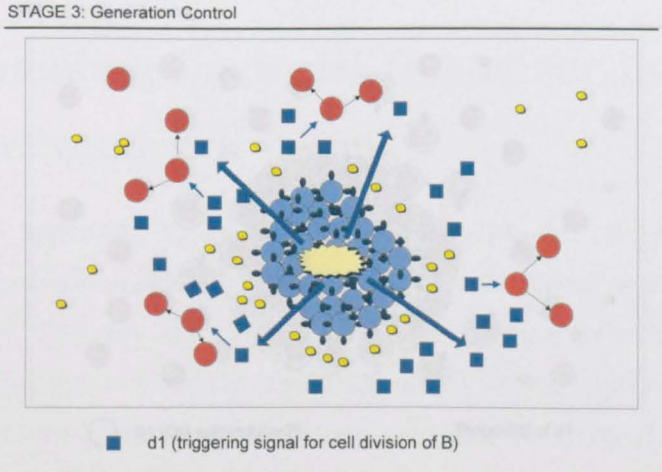


Figure 2.4: Quorum sensing (Step III)

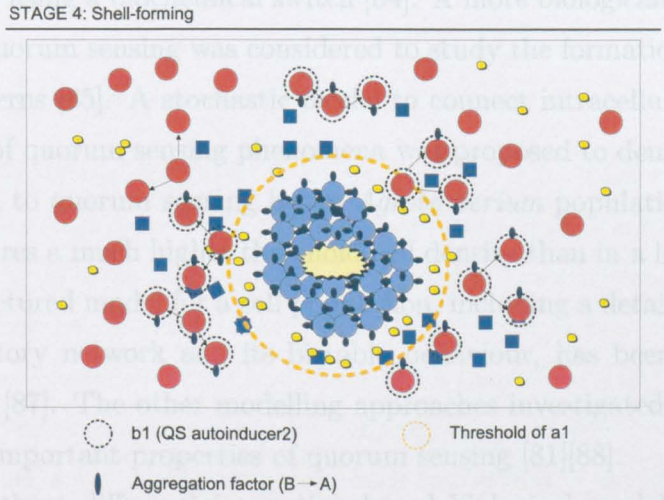


Figure 2.5: Quorum sensing (Step IV Part 1)

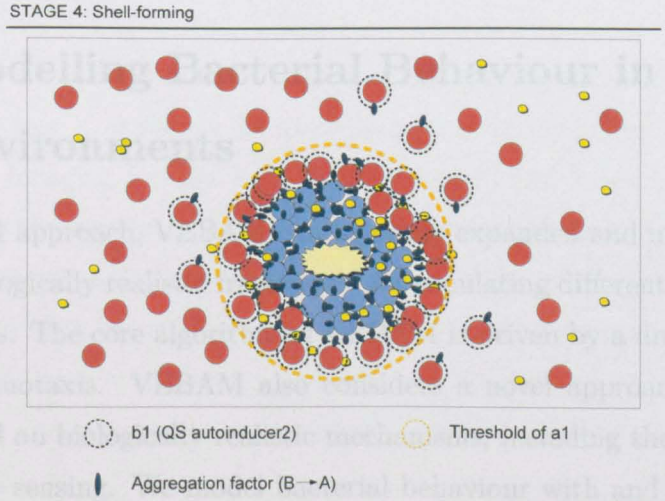


Figure 2.6: Quorum sensing (Step IV Part 2)

quorum sensing in *Pseudomonas.aeruginosa* is presented to show how quorum sensing works using a biochemical switch [84]. A more biologically realistic description of quorum sensing was considered to study the formation of bacterial foraging patterns [85]. A stochastic model to connect intracellular and population scales of quorum sensing phenomena was proposed to demonstrate that the transition to quorum sensing by an *Agrobacterium* population in a liquid medium requires a much higher threshold cell density than in a biofilm [86]. A spatially structured model for a cell population, including a detailed discussion of the regulatory network and its bistable behaviour, has been investigated and analysed [87]. The other modelling approaches investigated also revealed a number of important properties of quorum sensing [81][88].

However, these differential equation based biological models, introduced above for bacterial foraging patterns and/or quorum sensing, share the same drawbacks, such as, relying on experimental data. Moreover, they are not able to be generalised for analysis of various events within the biological system from which the data were collected. Therefore, it is rare that these models could be used to describe complex biological processes such as evolution.

2.5 Modelling Bacterial Behaviour in Varying Environments

As an IbM approach, VEBAM can be easily expanded and updated, while keeping a biologically realistic framework, for simulating different bacterial foraging patterns. The core algorithm of VEBAM is driven by a finely simulated model of chemotaxis. VEBAM also considers a novel approach to quorum sensing, based on biologically realistic mechanisms, including the employment of cell density sensing. We model bacterial behaviour with and without quorum sensing, and illustrate numerical differences between them acknowledging a wide variety of intrinsic and extrinsic properties of bacterial foraging behaviors. Population growth and nutrient variability influenced by quorum sensing, and the potential application of VEBAM are discussed. This model has been

tested in a varying environment, which demonstrates its potential for dynamic problem solving.

2.5.1 Architecture

The architecture of VEBAM is shown in Fig. 2.7. This model aims to investigate the foraging patterns of *E.coli* and colony development in a varying environment. It includes two fundamental elements: the environment and cells. The environment includes the properties of an artificial surface used for search, such as the characteristics of the boundary and a map of food distribution. The food distribution map is initialised by defining a specific function.

The cells are a set of bacteria, and each individual is assumed to have a receptor that detects the value of its surrounding environment. It also has a decision making system, determining the direction of movement, energy and status of cells, as well as controlling the quorum sensing switch according to information obtained by the receptors. These activities last during the cells' lifetime, enabling them to interact with the environment and then modify the food map. To simulate the quorum sensing phenomena, a cluster density detection scheme is introduced, which functions as a measure of the density of the cell population. As the cells grow independently from each other, the timescale presents a challenge in modelling such decentralised behavior. In VEBAM, the foraging process is artificially defined and fitted into time slots by introducing the concept of 'step'. The synchronisation unit is responsible for stepping the model in the time scale and keeps a global clock that synchronises the updating of both the environment and the cells. Updating the environment includes modification of nutrient distribution, while updating of cells consists of relocating them according to their positions and initialisation of newly created daughter cells.

One of the important aspects of this model is to emulate the bacterial chemotactic process, supported by a decision making system which determines the behavioural output of bacteria, according to variation in the environment. The behaviour of individual peritrichously flagellated bacteria can be described

in terms of run intervals, during which the cell swims approximately in a straight line, interspersed with tumbles during which the organism undergoes a random reorientation. If the concentration level of an attractant is increasing, a bacterium responds by reducing its tumbling probability. Thus the duration of run is longer when bacteria are swimming up a chemoattractant gradient [71]. Based on these activities, VEBAM adopts a decision making system that determines the direction of movement based on probability instead of only the random walk used in the description of bacterial behaviour in other IbMs [89].

During the chemotactic process, the states of cells change with interactions between the environment and these cells. The cells consume their energy to move and meanwhile they are rewarded by nutrient proportionally according to prevailing conditions. If the available nutrient for a cell is less than a certain amount, the cell gets all the available nutrient, and the nutrient is consumed fully in that location. If its energy falls below a critical level, the cell becomes inactive. In this case, the cell is motionless. Whilst the energy of a cell is higher than a pre-determined level, it can divide into two daughters, each of which becomes a new cell and a new life cycle begins. The detailed description of these elements is presented in Section 2.5.2.

VEBAM also provides the switching mechanism for quorum sensing. The population density of bacteria is assessed by detecting the concentration of a particular autoinducer, *i.e.* a signalling molecule, related to the density of a cluster of cells. At a certain stage of the bacterial foraging process, density would increase. A colony of bacteria would respond to the density by activating a specific gene expression program. This phenomenon, which provides intercellular communication, is exhibited commonly in Gram-negative bacteria, such as *E.coli*. The detailed description of this mechanism is given in Section 2.4.

2.5.2 Framework

A mathematical framework has been developed in this thesis to simulate the VEBAM. In the framework, the components of this model, including environment, cells, the synchronisation and interaction between them and population

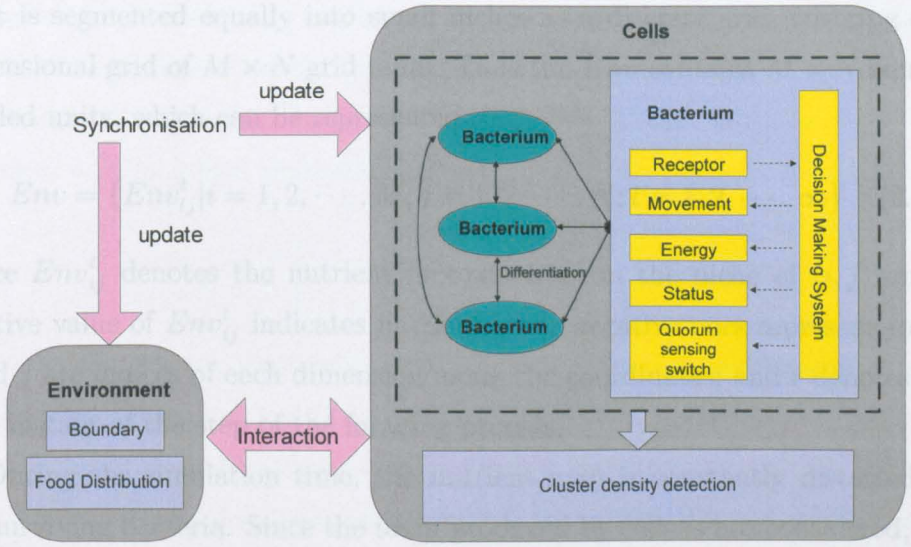


Figure 2.7: VEBAM architecture

growth, will be mathematically described.

Environment

The model is set in an artificial environment for bacterial movements. This environment contains potential nutrients and toxins. The former provides the energy required by bacteria to survive, while the latter may be inimical to bacterial activity. An environment includes the properties of an artificial surface used for search, such as the characteristic of the boundary and the map of food distribution. The food distribution map is initialised by defining a specific energy distribution function. The interactions between the environment and cells are updated during the computation process, in which cells are moving to obtain energy. After each step of cells' movements, the nutrient distribution is modified in relevant grid units.

The environment is defined as

$$Env[f_{\mathrm{n}}, \beta] \tag{2.5.1}$$

where f_{n} indicates the nutrient distribution and β represents the boundary characteristic, indicating whether it is reflective or periodic. The environ-

ment is segmented equally into small niches as a discrete grid, covering a 2-dimensional grid of $M \times N$ grid units. Thus the Env contains $M \times N$ equally divided units, which can be represented as:

$$Env = \{Env_{ij}^t | i = 1, 2, \cdots, M; j = 1, 2, \cdots, N; t = 1, 2, \cdots, \infty\} \quad (2.5.2)$$

where Env_{ij}^t denotes the nutrient or toxin level in the niche of (i, j) and a positive value of Env_{ij}^t indicates nutrient while negative ones represent toxin; i and j are indices of each dimension along the coordinates; and t denotes the time instant of the step of the foraging process.

During the simulation time, the nutrient map is constantly distorted by the surviving bacteria. Since the toxin produced by cells is not considered, the only factor that modifies the nutrient map is the consumption of nutrients by bacteria. Thus, the nutrient depletion in the environment can be expressed as

$$Env_{ij}^{t+1} = Env_{ij}^t - \sum_{l=1}^{P_{ij}} C_{\text{gain}}^l \quad (2.5.3)$$

where C_{gain}^l is the nutrient obtained by the l^{th} bacterium in grid (i, j) per step. P_{ij} is the number of bacteria in grid (i, j) at time t.

Bacteria and their foraging patterns

Following the definition of the environment, a set of bacteria is represented as:

$$B = \{B_l^t | l = 1, 2, \cdots, P; t = 1, 2, \cdots, T\} \quad (2.5.4)$$

where

$$B_l^t = [p_l^t, \varepsilon_l^t, \varphi_l^t, \lambda_l^t, \tau_l^t, \delta_l^t, \phi_l^t] \quad (2.5.5)$$

denotes an individual bacterium or cell; P is the maximum population size in the foraging process and T the final time of this process; B_l^t has its own position p_l^t, $p_l^t = \{i, j | l, t\}$, energy ε_l^t, and status φ_l^t, $\varphi_l^t \in \{\Phi_{\text{I}}, \Phi_{\text{A}}, \Phi_{\text{D}}\}$, where $\Phi_{\text{I}}, \Phi_{\text{A}}, \Phi_{\text{D}}$ stand for the status of Inactive, Active and Divided, respectively. An action τ_l^t depends on a probability-based variable λ_l^t to indicate whether the cell is running or tumbling at time t, $\tau_l^{t+1} \in \{\chi_{\text{R}}, \chi_{\text{T}}\}$, where χ_{R} denotes

the action of *Run*, while χ_{T} represents *Tumble*. If it is tumbling, the cell moves with an orientation ϕ_l^t as an angle formed by the flagellar axis. For the run orientation, ϕ_l^t is the same as ϕ_l^{t-1}; δ_l^t is the length of run, which is set to be a constant; and λ_l^t, which represents a probability of run, is defined as follows:

$$\lambda_l^{t+1} = \begin{cases} \Lambda, & : \text{ if } Env_{ij}^t < Env_{ij}^{t^*} \\ 1-\Lambda, & : \text{ if } Env_{ij}^t \geq Env_{ij}^{t^*} \end{cases} \tag{2.5.6}$$

where Λ represent the probability of run, which is determined according to the change of environment, and $0 < \Lambda < 0.5$.

It is known that chemotactic cells tumble less frequently if the cells experience an increase in chemoattractant concentration over a period of time. Equation (2.5.6) implies that a better environment leads to a higher probability of run. However in nature the direction of bacterial motion depends upon the changes of chemoattractant concentration levels, which take place at the cell site over a time interval of up to 4 seconds [76]. VEBAM considers the bacterial chemoattractant concentration over a period of time, for example values of the environment observed over the past three steps, as shown in Fig. 2.8. In this case, Env_{ij}^{t-3}, Env_{ij}^{t-2} and Env_{ij}^{t-1}, are adopted to obtain an average measure of the previous environment, which is denoted by $Env_{ij}^{t^*}$, where t^* indicates the fictitious time instant associated with the average value of the changed environment. Bacteria compare their receptor occupancies between those which occurred at t^* and t, *i.e.* let $d = Env_{ij}^t - Env_{ij}^{t^*}$. If $d < 0$, the cell will reduce the rate of run to Λ, otherwise it remains at $1 - \Lambda$, as described in (2.5.6). It should be mentioned that (2.5.6) influences whether the next action of the bacterium is *tumble* or *run*. Given a random number $\zeta \in (0, 1)$, the decision is made by the following equation:

$$\tau_l^{t+1} = \begin{cases} \chi_{\mathrm{R}}, & : \text{ if } \zeta > \lambda_l^{t+1} \\ \chi_{\mathrm{T}}, & : \text{ if } \zeta \leq \lambda_l^{t+1} \end{cases} \tag{2.5.7}$$

Hence, the angle ϕ_l^t can be given as follows:

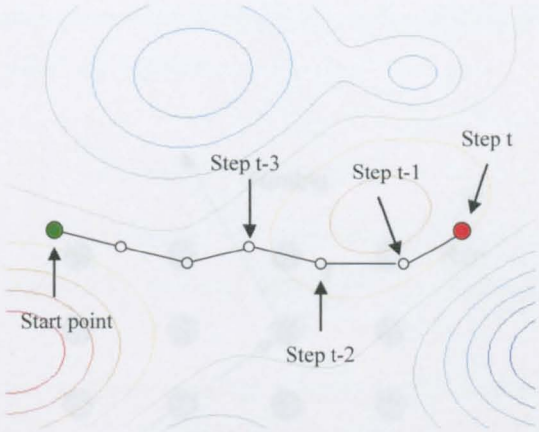


Figure 2.8: Environmental comparison

$$\phi_l^{t+1} = \begin{cases} \phi_l^t, & : \text{ if } \tau_l^{t+1} = \chi_{\mathrm{R}} \\ \phi_l^t + \theta, & : \text{ if } \tau_l^{t+1} = \chi_{\mathrm{T}} \end{cases} \tag{2.5.8}$$

where θ is given as

$$\theta = 2\arcsin(2\mu) \tag{2.5.9}$$

where μ is a random number, and $\mu \in [-0.5, 0.5]$. The nonlinear relationship between θ and μ is shown in Fig. 2.10.

If $\tau_l^{t+1} = \chi_{\mathrm{T}}$, the tumbling angle of *E.coli* is randomly generated and non-uniformly distributed. It is found that the mean tumbling angle of wild type *E.coli* is about 68° with a standard deviation of 36° approximately [6]. In this case, the tumbling angle for the next step is the previous angle plus θ, as shown in Fig. 2.9. Theoretically, θ could be any angle between $-\pi$ and π.

It is known that bacteria swim by rotating whip-like flagella driven by a reversible motor embedded in the cell wall. The direction of flagellar rotation determines whether peritrichously flagellated cells run or tumble. For *E.coli*, the *run* and *tumble* intervals are exponentially distributed approximately with means of 1 second and 0.1 second respectively in an isotropic medium [76]. Since the duration of the activity of *run* is about 10 times of the activity of *tumble*, the time taken by *tumble* is ignored in this model. The angle changes

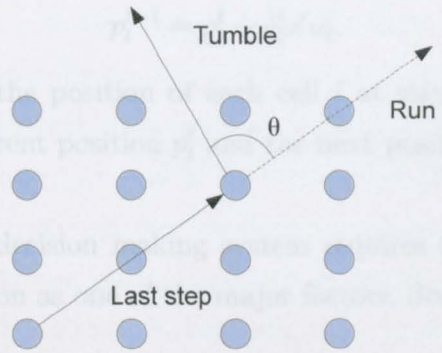


Figure 2.9: The directions of tumble and run

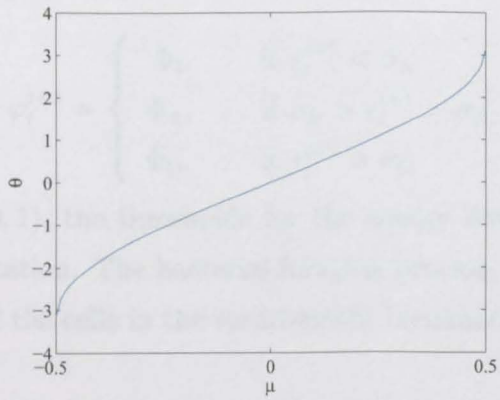


Figure 2.10: The relationship between μ and θ

while the *tumble* takes place. As a result, the cell position changes in each step, which can be described as follows:

$$p_l^{t+1} = p_l^t + \delta_l^t \angle \phi_l^t \tag{2.5.10}$$

where p_l^{t+1} indicates the position of each cell l at step $t+1$. The distance between the cell's current position p_l^t and the next position p_l^{t+1} can be represented by $\delta_l^t \angle \phi_l^t$.

At this step, the decision making system requires the cell statistics that include energy variation as one of the major factors, described as follows:

$$\varepsilon_l^{t+1} = \varepsilon_l^t + \min(Env_{ij}^{t+1}, C_{\text{gain}}^l) \tag{2.5.11}$$

where $(i,j) = p_l^{t+1}$. In each step, if the nutrient available in grid (i,j), Env_{ij}^{t+1}, is less than the nutrient required by the cell, C_{gain}^l, the cell lowers its nutrient consumption level and then takes up the available nutrient in this grid.

The decision making system then decides whether it will be alive or dead based on the statistics of each cell. A cell can either carry on living or be removed from the system or divide into two cells according to the following equation:

$$\varphi_l^{t+1} = \begin{cases} \Phi_{\text{I}}, & : \text{ if } \varepsilon_l^{t+1} < \sigma_{\text{A}} \\ \Phi_{\text{A}}, & : \text{ if } \sigma_{\text{D}} > \varepsilon_l^{t+1} > \sigma_{\text{A}} \\ \Phi_{\text{D}}, & : \text{ if } \varepsilon_l^{t+1} > \sigma_{\text{D}} \end{cases} \tag{2.5.12}$$

where $[\sigma_{\text{D}},\sigma_{\text{A}}] \in (0,1)$, the thresholds for the energy level, are preset at the start of the computation. The bacterial foraging process, as addressed above, is repeated until all the cells in the environment terminate.

2.5.3 Proliferation of the population

Since motility of *E.coli* cells in clusters is formed by chemotactic aggregation, the population of *E.coli* in clusters varies according to the available nutrient. Given good growth conditions, a bacterium grows in size until a new

septal cell wall grows through the centre of the cell to produce two daughter cells. In favorable conditions, the generation time can be as low as 20 minutes. Generally, the population growth has three phases, namely a lag phase, an exponential phase and a death phase. In the lag phase, growth is slow at first, while the cells acclimatise to the food in their new habitat. It was discovered that the growth rate doubles in the middle of the division cycle, which leads to a doubling of the reproduction rate of the population [90]. In the death phase, as more and more bacteria compete for dwindling food, growth stops and the number of bacteria becomes stable until toxic waste products build up; at the same time food is depleted and the bacteria begin to die.

Cell populations have been modelled using partial differential equations, gas kinetic theory, cellular automata, and Brownian agents [91] [92] [93]. These models, however, either concentrate on macroscopic calculation of the process without a detailed description of the internal mechanisms, or apply a specific individual behaviour rather than chemotaxis.

VEBAM has adjustable growth and reproduction rates, which can be determined dynamically according to environmental variation. During a cell's lifetime, mass grows as long as the cell obtains energy from the environment. When cellular energy exceeds a preset threshold, it divides into two daughters, each of which gets a randomised allocation of the initial energy, as shown in Fig. 2.11. In a rich nutrient area, a cell takes a shorter time to obtain enough energy for reproduction. The population variation can be represented as:

$$P^{t+1} = P^t + \Delta P_{\mathrm{d}} - \Delta P_{\mathrm{i}} \tag{2.5.13}$$

where ΔP_{d} and ΔP_{i} indicate the populations of divided cells and inactive cells, respectively. They are two main factors affecting variation in the population. After each step of the chemotactic process, the divided cells are listed in a memory pool to keep a record of the evolutionary trajectory, while the inactive cells are deleted from the memory in the simulation study.

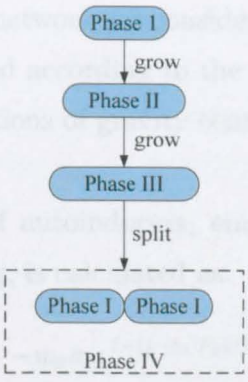


Figure 2.11: Division process of a cell

2.6 Simulation Studies

2.6.1 The definitions and computation of quorum sensing in VEBAM

In VEBAM, quorum sensing is calculated based on the definition and computation rules of clusters. A set of clusters is an aggregation of population, and is defined as follows:

$$C = \{C_k | k \leq P\} \tag{2.6.1}$$

where

$$C_k = [D_k, G_k, F_k] \tag{2.6.2}$$

denotes the k^{th} cluster in the population, D_k the population density, G_k the position of the centre of gravity, and F_k the diffusion exponent of the autoinducers in cluster C_k.

At each chemotactic step, bacteria move from one position to another, and they tend to move towards a nutrient-rich area. In general, variation in bacterial population density is affected by four factors: the availability of nutrients, the presence of competitors, the host response and the quorum sensing network. However, due to the complex relationship of these four factors and also insufficient data used for simulation studies, only the availability of the nutri-

ents and the quorum sensing network are considered. In VEBAM, each cluster, C_k, the population is classified according to the distribution of cells and their population density. The positions of gravity center, G_k, are also calculated for each cluster C_k.

At time t, the amount of autoinducers, emitted by cluster C_k with the mechanism of quorum sensing, is calculated as:

$$A^t_{k,i,j} = -w_k a_p e^{(-(t-t_0)F_k \| G_k - p_{i,j} \|_2^2)} \tag{2.6.3}$$

where w_k is the weight of cluster C_k, t_0 is the time at which the quorum sensing is turned to the 'on' state, a_p denotes the amplitude of the peak of the autoinducer congregation (and is to be tuned as appropriate) and $p_{i,j}$ covers the whole environment.

The autoinducers could be added to the environment as an attractive factor. Then the environment is modified as

$$Env^{t+1}_{ij} = Env^t_{ij} - \sum_{l=1}^{P_{i,j}} C^l_{\text{gain}} + A^t_{k,i,j} \tag{2.6.4}$$

The energy of each bacterium producing autoinducers also needs recalculation. Select $C_m \subset C$, where C_m is the cluster which produces autoinducers, the energy of the l^{th} individual in C_m is modified as:

$$\varepsilon^{t+1}_l = (1-\alpha)\varepsilon^t_l + \min(Env^{t+1}_{ij}, C^l_{\text{gain}}) \tag{2.6.5}$$

where $l \in [1, 2, \cdots, n_m]$, n_m is the total number of individuals in cluster m and α is the coefficient used to control the rate of the energy to autoinducer transformation.

To model the bacterial foraging behaviour, a software package has been developed, based on the MATLAB language. It involves three major parts: initialisation, the chemotaxis step and colony updating. The pseudo code of the whole algorithm is shown in Table 2.1.

Table 2.1: Pseudo code for VEBAM

Randomly initialise states of bacteria represented by (2.5.2) and (2.5.5);
Tumble: For bacterium l, the initial direction is randomly chosen;
While (Number of survival bacteria > 0)
FOR (Chemotactic step t)

Environment:	Without quorum sensing, Env_{ij}^{t+1} is updated using (2.5.3), as the cells are now ready to run in this cycle.
FOR (bacterium l)	
Decision making:	Get the environmental value of each bacterium l as $Env_{p_l^t}^t$; the average environmental value of the past three steps, $Env_{p_l^t}^{t^*}$ is stored in the cell's memory for making decisions. Calculate λ_l^{t+1} by (2.5.6) and τ_l^{t+1} using (2.5.7). If $\tau_l^t = \chi_R$, **Run**; otherwise, **Tumble**;
Tumble:	Calculate the tumbling angle by (2.5.9). Bacterium l moves to a new position using (2.5.10);
Run:	For bacterium l, it takes another unit walk of the same direction by (2.5.10).
END FOR	
Individual:	Calculate the energy of bacteria using (2.5.11). The decision making system determines the status of cells by (2.5.12).
Colony updating:	Using statistics and chemotactic rates obtained from Table 2.2, modify the environmental value of the cell's position, and calculate the current population by (2.5.13); the positions of divided cells are recorded.

Quorum sensing: Once the density of a population reaches the threshold, the quorum sensing switch turns on. The clusters C_k of the population are classified using the density-based algorithm; calculate each element of C_k defined in (2.6.2), and the autoinducer by (2.6.3); modify the environment according to quorum sensing and the energy of the autoinducer-generating bacteria, by (2.6.4) and (2.6.5), respectively.

END FOR
END While

2.6.2 Environment setting and parameter selection

The VEBAM is set in a multi-modal environment. The nutrient distribution of the toroidal grid environment at $t = 0$ is represented by the function $f_{\mathrm{n}}(x, y)$ given by [69]:

$$\begin{aligned} f_{\mathrm{n}}(x,y) &= 5e^{-0.1((x-15)^2+(y-20)^2)} - 2e^{-0.08((x-20)^2+(y-15)^2)} \\ &+ 3e^{-0.08((x-25)^2+(y-10)^2)} + 2e^{-0.1((x-10)^2+(y-10)^2)} \\ &- 2e^{-0.5((x-5)^2+(y-10)^2)} - 4e^{-0.1((x-15)^2+(y-5)^2)} \\ &- 2e^{-0.5((x-8)^2+(y-25)^2)} - 2e^{-0.5((x-21)^2+(y-25)^2)} \\ &+ 2e^{-0.5((x-25)^2+(y-16)^2)} + 2e^{-0.5((x-5)^2+(y-14)^2)} \end{aligned} \tag{2.6.6}$$

The above equation is normalised as $\bar{f}_{\mathrm{n}}(x, y) = f_{\mathrm{n}}(x, y)/f_{\max}$, where $f_{\max} = \max(f_{\mathrm{n}}(x, y))$, $x, y \in [0, 30]$, both x and y change with an incremental value of 0.1 and the environment is divided into 300 × 300 grids.

During the whole process of computation, the nutrient in the environment is constantly consumed by bacteria at each grid unit, which they occupy until the nutrient is lower than a minimal level. The bacteria are initially located

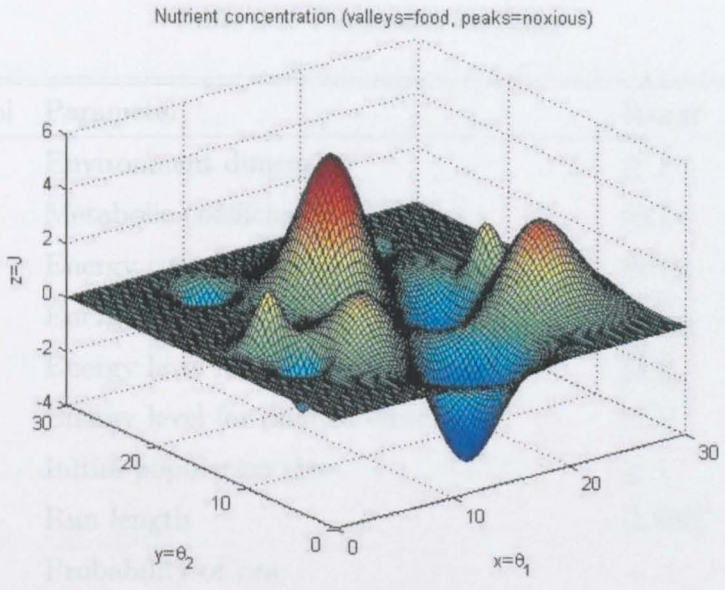


Figure 2.12: Environment setting

in a distribution generated by the Halton Quasi-random sequence, which has been shown to assist in reducing generations needed in GAs effectively [94]. The initial population size is 100. The parameter settings are given in Table 2.2.

2.6.3 Density-based clustering algorithm

As discussed in Section 2.4, in each cluster, C_k, the population can be classified, according to the positions of the cells, using the method of 'density-based spatial clustering' [95]. If we define an individual as a 'point', then all points can be classified as follows:

First of all, ϵ, the neighbourhood radius of a cluster is defined. Given the parameters: r, the minimal number of points in a cluster; X, $X = \{x_l | l = 1, 2, \cdots, P\}$, a set of the positions of all points; and n, the dimension of the

Table 2.2: Parameter settings

Symbol	Parameter	Range	Value
Env	Environment dimension	≥ 1	2
E_g	Metabolic coefficient	(0,1)	0.02
E_c	Energy cost for a swim	(0,1)	0.01
E	Energy level of organism	[0,1]	N/A
σ_A	Energy level for active state	[0,1]	0.2
σ_D	Energy level for divided state	[0,1]	0.8
P_I	Initial population size	≥ 1	100
δ	Run length	[1,100]	5
Λ	Probability of run		
	when Env_{cur} is worse than Env_{pre}	[0,1]	0.3
a_{p}	Amplitude of the peak in quorum sensing	[0,1]	0.1
F_k	Diffusion effector	[0,10]	0.5
α	Metabolism factor for quorum sensing	[0,1]	0.05

positions, the ϵ is calculated as:

$$\epsilon = \left(\prod_{d=1}^{n} (\max(X_d) - \min(X_d)) \times k \times \Gamma(a))/M \right)^{1/n} \tag{2.6.7}$$

where $\Gamma(a) = \int_0^\infty e^{-t} t^{a-1} dt$, $a = 0.5n + 1$, $M = r(\pi)^{n/2}$, and $X_d = \{x_{ld} | l = 1, 2, \cdots, P\}$ and x_{ld} is the position value on the d^{th} dimension.

Given an arbitrary point p in the environment, let d_i be the distance from p to its i^{th} nearest neighbour; set D as the radius of the neighbourhood of p, and the D-neighbourhood of p contains exactly $s + 1$ points, where s is the total number of neighbours of p. In this neighbourhood, the longest distance exists, denoted as $d_{\max}$, $d_{\max} = \max_i\{d_i\}$. If and only if $d_{\max} \leq \epsilon$ and $s + 1 \geq r$, the D-neighbourhood of p is considered to be a cluster.

For each cluster C_k, all points of C_k fall into two categories: points inside of the cluster (core points) and points on the border of the cluster (border points). Given a point p, which belongs to C_k, if $d_{\max} \leq \epsilon$, it is called a core

point, p_c, otherwise it is deemed a border point, p_b.

The position of the gravity center G_k, for each cluster C_k, is obtained according to the identification of core points. Set the position of a p_c in C_k as P_{kc}, and $P_{kc} = [P_{kc,1}, P_{kc,2}, \cdots, P_{kc,d}, \cdots, P_{kc,n}]$, then the coordinate of G_k in the d^{th} dimension is calculated as:

$$G_{kd} = \frac{\sum_N P_{kc,d}}{N} \tag{2.6.8}$$

where N is the number of core points in cluster C_k.

2.6.4 Simulation results and analysis

Single bacterial behaviour

While foraging, bacteria absorb energy from nutrients in the environment in order to survive and reproduce. In this case, the nutrient is depleted constantly until the computation ends. Meanwhile, in accordance with the principle of chemotaxis, bacteria are always moving towards a richer nutrient area, which in turn, modifies the distribution of nutrient. In this way, the environment changes following each iteration of bacterial movement.

Figures 2.13~2.16 illustrate the trajectories of bacteria in two separate experiments. In VEBAM, σ_D is set to 0.8 for a cell to split; and σ_A is set to 0.2 for a cell to become active. The probabilities for action of *Run* when the environment is worse is set to $\Lambda = 0.3$. The run length is set to 5 grid units per step. The energy gained and the energy consumed by each cell at a single step are denoted by E_g and E_c respectively. Figures 2.13 and 2.14 show the behaviour of cells 4 and 18, where $E_g = 0.2$ and $E_c = 0.1$. Figures 2.15 and 2.16 concern cells 8 and 14, where $E_g = 0.02$ and $E_c = 0.01$ are adopted.

Figure 2.13 shows that when a bacterium moves up a gradient, it approaches an area that contains higher energy. The action of *tumble* and *run* are combined together to execute the chemotactic process. However, the cell cannot stay still in the area containing the highest energy because of the stochastic nature of its movement. The cell continues to run across this area before a tumble takes place, as shown in Fig. 2.13. This foraging behavior could be

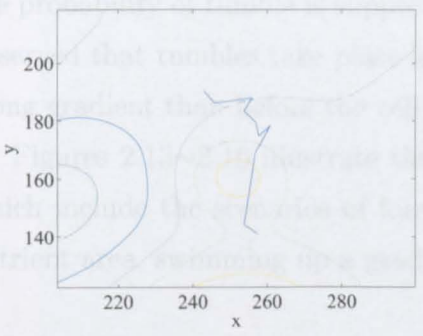


Figure 2.13: Cell 4 ($E_g = 0.2$, $E_c = 0.1$)

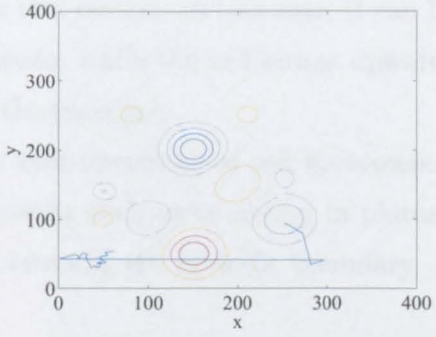


Figure 2.14: Cell 18 ($E_g = 0.2$, $E_c = 0.1$)

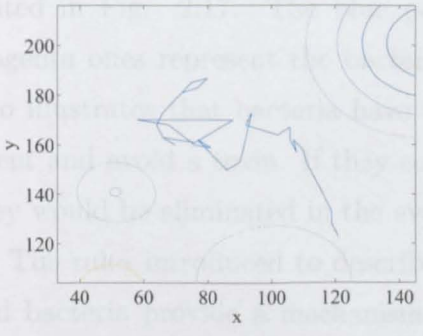


Figure 2.15: Cell 8 ($E_g = 0.02$, $E_c = 0.01$)

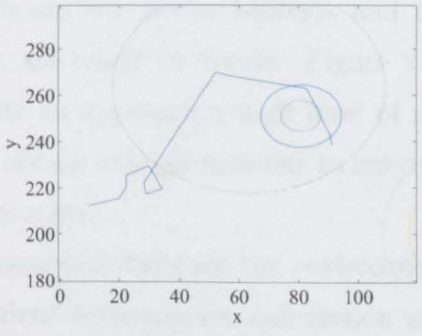


Figure 2.16: Cell 14 ($E_g = 0.02$, $E_c = 0.01$)

considered as turbulence, which enables the cell to avoid getting trapped in the local optima. Figure 2.14 shows that a bacterium starts at a position around $[50, 50]$, with a number of tumbles until it reaches the boundary where $x = 0$. The cell continues to move in the same direction. For the simulation purpose, considering the periodical boundary, the cell is arranged to re-enter the environment from the right side, across the boundary which is indicated by the dashed line. After re-entering the environment, the cell moves up the gradient towards the area of richest nutrient. A pure random walk could be observed in Fig. 2.15, from which it can be seen that before a cell detects a gradient change, it chooses a next action randomly in the environment. This phenomenon indicates that a foraging cell moves without a fixed direction in a plateau nutrient area. Meanwhile, Figure 2.16 illustrates that the cell takes a probability-based run/tumble strategy to approach the nutrient-rich area, and

the probability of tumble is suppressed over this period. In this case, it can be observed that tumbles take place less frequently when the cell swims upwards along gradient than before the cell detects the gradient.

Figures 2.13~2.16 illustrate the various characteristics of cell movements, which include the scenarios of foraging patterns such as tumbling in plateau nutrient area, swimming up a gradient and crossing the periodic boundary.

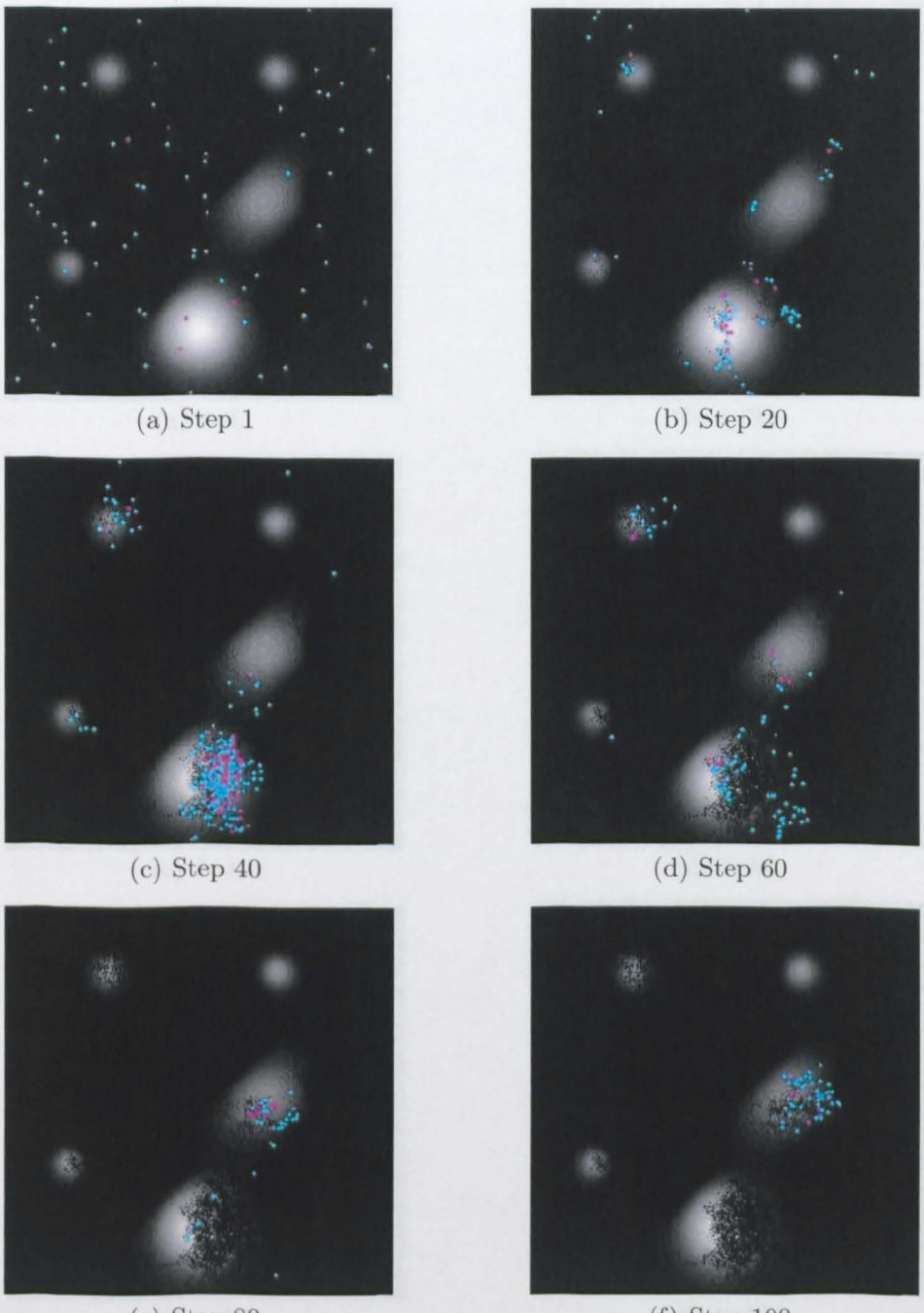
2.6.5 Population evolution without quorum sensing

The population evolution of VEBAM has been simulated and it is illustrated in Fig. 2.17. The blue points indicate the active bacteria and the magenta ones represent the bacteria which are ready to divide. Figure 2.17 also illustrates that bacteria have the ability to approach a high level of nutrient and avoid a toxin. If they could not obtain enough nutrient to survive, they would be eliminated in the evolution process.

The rules introduced to describe the interaction between the environment and bacteria provide a mechanism for nutrient consumption calculation and bacterial status updating. The bacterial swimming mechanism is a combination of rules and stochastic searching. Furthermore, the population modelling includes the processes of initialisation, reproduction and termination of bacteria. VEBAM is a generic and expandable IbM for simulating population evolution based on the framework for modelling bacterial foraging patterns.

2.6.6 Population evolution with quorum sensing

VEBAM follows the basic biological rules by which each cell attempts to obtain as much energy as possible in a relatively short period of time. The population evolution of the VEBAM has been simulated and it is illustrated in Fig. 2.18. The evolution process proceeds via 100 steps, featuring the swarming of the cells towards the high-level nutrient as shown in Fig. 2.18. It has been noted in Fig. 2.18(b) that the quorum sensing was switched 'on' at or before $t = 20$, as four clusters were identified in this case (using (2.6.1)). From this

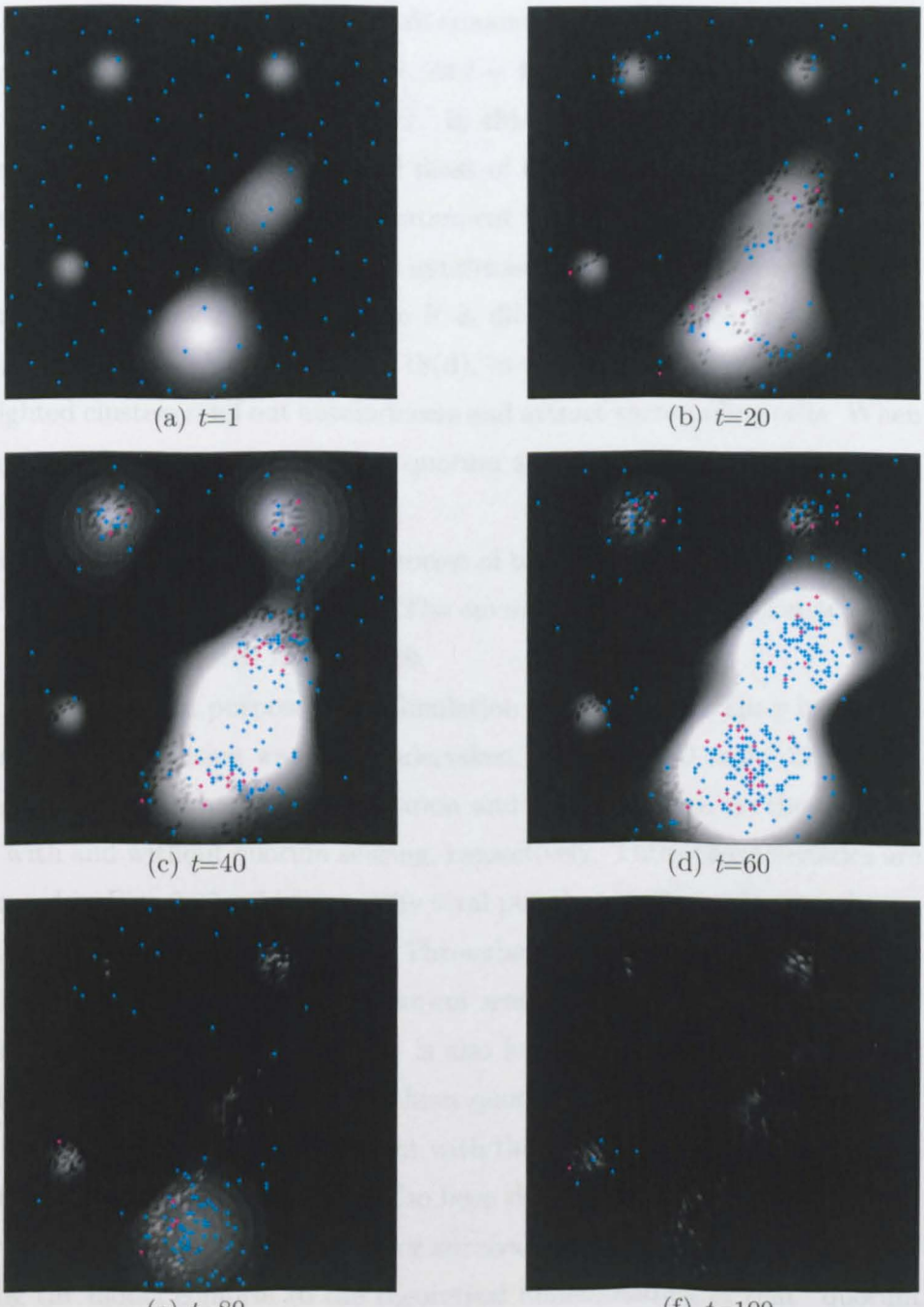


(a) Step 1 (b) Step 20

(c) Step 40 (d) Step 60

(e) Step 80 (f) Step 100

Figure 2.17: The interaction between cells and environment in 100 steps



(a) t=1 (b) t=20

(c) t=40 (d) t=60

(e) t=80 (f) t=100

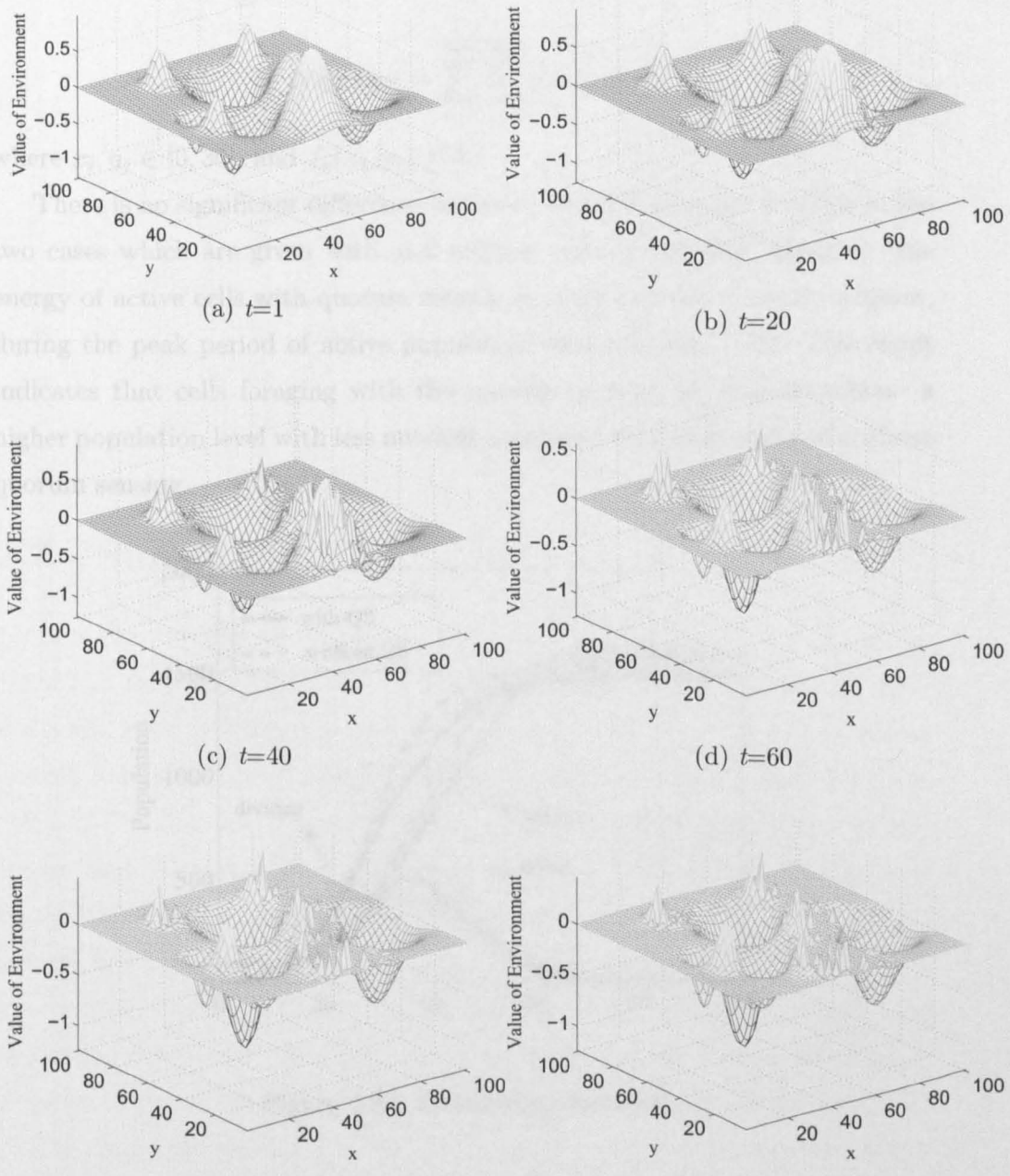
Figure 2.18: The interaction between cells and environment in 100 steps

time, the cell steadily produced a small amount of autoinducer molecules, that can freely diffuse in and out of the cell. At $t = 40$, three clusters were identified, which are highlighted in Fig. 2.18(c). In this case, the population densities of the top two clusters were low and most of the autoinducer molecules were washed out and dispersed in the environment by diffusion. In other words, at a low cell density the autoinducer is synthesised at a basal level and diffuses into the surrounding medium, where it is diluted. Much clearer features of quorum sensing are shown in Fig. 2.18(d), in which the cells in the two most highlighted clusters send out autoinducers and attract surrounding cells. When the nutrient is consumed gradually, quorum sensing no longer functions and active cells do not affect each other.

Figure 2.18 shows the evolution process of bacteria with autoinducers artificially described in the environment. The environmental modification achieved using (2.6.4) is illustrated in Fig. 2.19.

For comparisonal purposes, the simulation of bacterial foraging behaviour without quorum sensing was also undertaken. Figures 2.20 and 2.21 demonstrate differences in population evolution and their influence on the environment with and without quorum sensing, respectively. Three characteristics are illustrated in Fig. 2.20, which are: the total population, the active population and the inactive population of cells. Throughout the process of 100 steps, the total population in the case with quorum sensing is lower than that without it. However, the inactive population is also lower and the peak of the active population is higher than the one without quorum sensing. The simulation results shown in Fig. 2.20 are consistent with the experimental data in terms of population-time profile [96]. It has also been demonstrated that quorum sensing can optimise population growth or survival which is an important step in making the model conform to the theoretical understanding of that "quorum sensing provides a benefit at the population level by increasing the production of cooperative exoproducts that can aid growth in certain environmental conditions" [97].

The available nutrient in the environment and that obtained by active and



(a) t=1 (b) t=20

(c) t=40 (d) t=60

(e) t=80 (f) t=100

Figure 2.19: Environmental change in 100 steps

inactive cells are shown in Fig. 2.21. The results were calculated using the following equation:

$$Nutrient = \sum_{i=1}^{300}\sum_{j=1}^{300} f_{\mathrm{n}}(x_i, y_j)$$

where $x_i, y_j \in [0, 30]$, and $f_{\mathrm{n}}(x_i, y_j) \geq 0$.

There is no significant difference between the total available nutrient in the two cases which are given with and without quorum sensing. However, the energy of active cells with quorum sensing is much less than the one without, during the peak period of active population shown in Fig. 2.20. This result indicates that cells foraging with the quorum sensing are able to achieve a higher population level with less nutrient consumed than that achieved without quorum sensing.

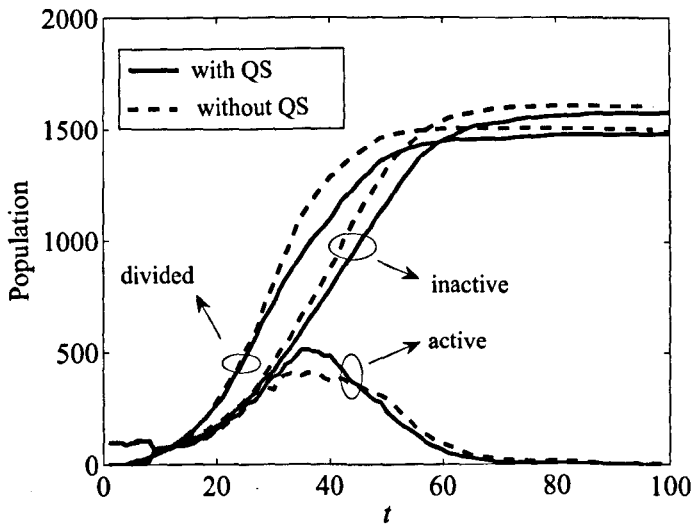


Figure 2.20: Population evolution

Further simulation results on quorum sensing influence can be obtained if the model parameters, such as λ and F_k, are given in different values. However, the basic features of population and environment variation would not change. Therefore, we do not present further results on this aspect in this chapter.

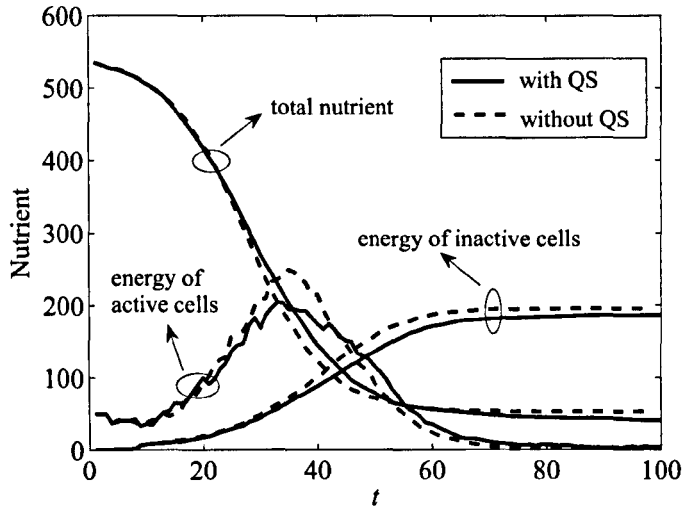


Figure 2.21: Energy distribution

2.7 Summary

This chapter has presented VEBAM for modelling bacterial foraging behaviour in varying environments. The details of individual foraging behaviour and population development process have been described. The chemotactic process, developed as a kernel of IbM from the novel model, has been evaluated in the simulation studies which include modelling a group of bacteria during an infinite number of generations. The quorum sensing mechanism has been successfully incorporated into this model and also evaluated in the simulation studies. A density-based clustering algorithm has been adopted to classify the clusters of cells, which assists in identifying the timing of quorum sensing switching. The centre of gravity in each cluster has also been identified to pinpoint to centre for sending and diffusing autoinducers in the environment.

The simulation results show that the bacteria move towards nutrient, avoiding toxin in a chemotactic manner and this IbM-based approach is able to provide a plausible methodology to simulate the bacterial foraging behaviour. It has also been demonstrated that quorum sensing can significantly influence the variation of bacteria populations and environmental nutrients, which provides

a more biologically-realistic solution for modelling bacterial foraging patterns. Furthermore, the work presented in this chapter would provide a basis for the development of novel bacteria-inspired optimisation algorithms.

Chapter 3

Advances in Bacteria-inspired Algorithms

3.1 Introduction

The use of biological processes or behaviour as metaphor, inspiration, or enabler in developing new computing technologies has been emerging in recent years. These techniques are the so-called BIA. The main steps of BIA can be divided into two parts: 1) observing animal and human behaviour and modelling biological phenomena; 2) mimicking acquired knowledge to develop computing algorithms for solving the problems of engineering systems and machines.

Nature is a powerful paradigm. A number of BIAs have been reported, such as EAs stemmed from evolutionary theory; PSO inspired by flocking birds as well as other SIs with insects as its counterpart in nature. In comparison with the conventional gradient-based algorithms, BIAs, which can be self-tuned using raw experimental data, may require little or no knowledge of the physical system they emulate.

In Chapter 2, we have introduced the fundamental work concerning the modelling of bacterial foraging patterns. This modelling work has exhibited a potential in solving optimisation problems. However, as we understand, there is a huge gap between modelling and optimisation. Modelling is 'a study of a

miniature of the actual', thus it is concerned with representing the behaviour of bacterial foraging; while optimisation is generally problem-oriented, and does not require a strict biological description for each element. Therefore, analysing the nature of optimisation problems and developing the algorithms to solve these problems accordingly is investigated in this chapter.

In this chapter, three algorithms inspired by bacterial foraging patterns are described. The first one, BFAVP, is derived directly from VEBAM, the compnents of BFAVP are introduced in detail in comparison with their counterpart in VEBAM. To enhance the group behaviour of BaIAs, a modified algorithm, BSA is presented, which also adopts the merits of VEBAM. Finally, to reduce the computational load, another algorithm, PBO, is investigated.

3.2 From Models to Algorithms

Developing a system model always depends on the understanding of the phenomena of a physical system and the mathematical expressions available to represent the system. From the viewpoint of mathematics, chaotic dynamics is a typical complex system in which the order of mathematical equations increases exponentially as the system dynamics is described at a further level. However, in most industrial and public systems, the complexity of a system arises from its hierarchical structure, distributed local modes, hybrid dynamics, multiple mixed variables, stochastic nature, uncertainties and inconsistent component behaviour, *etc.* For these complex systems, there have been no established modelling approaches reported apart from dedicated methods applied to specific problems. To develop a new paradigm of complex system modelling, the study of biological systems, in particular at the bacteria level, would indicate a new direction to success. The use of biological processes or behaviour as metaphor, inspiration, or enabler in developing new computing technologies is emerging in the recent years. As aforementioned, IbM approaches have shown their success in modelling biological systems which could be more complex than any industrial systems. We are motivated, by modelling bacterial foraging pat-

terns, to develop a new systematic methodology to model not only complex biological systems but also non-biology systems. However, this goal requires a significant amount of efforts which are much beyond the work presented in this thesis.

On the other hand, it has been well understood that all BIAs built on modelling of genetic evolution, biological systems or animal behaviour and these modelling-based methods have been well developed for optimisation purposes. Currently, the complex while organised activities exhibited in bacterial foraging patterns could inspire approaches to solve complex optimisation problems. Therefore, we will focus on the study of major features of bacterial foraging patterns, built on modelling of these complex patterns as presented in the previous chapter, to develop novel, more efficient and powerful algorithms for complex optimisation problems.

3.3 Previous Studies of BaIAs

Not surprisingly, the study of bacterial foraging behaviour has become a new branch in the area of BIAs. The complex while organised activities exhibited in bacterial foraging patterns could inspire a new approach to solve complex optimisation problems. The underlying mechanism of the surviving of bacteria, especially *E.coli* in a complex environment has been reported by researchers in the area of biological sciences [6]. These research outcomes have been adopted by the models simulating bacterial foraging patterns [98].

In the early part of this century, a few optimisation algorithms based on models of bacterial chemotaxis emerged. However, these biological models in their original form are mostly used to formulate simple optimisation algorithms. Thus, how to add more features to the basic algorithm in order to obtain an enhanced optimisation strategy is still an open question.

3.3.1 Preliminary studies of bacterial foraging patterns

Previously, there have been two attempts to work towards this goal. One is called the 'bacterial chemotaxis (BC)' algorithm, proposed by Müller *et al.* [99], and the other is entitled 'Bacterial Foraging Algorithm (BFA)' which was invented by Passino [69]. In this section, these two algorithms will be briefly introduced.

BC is the first algorithm in the literature considering bacterial foraging behaviour to solve optimisation problems. However, although it is now well recognised that bacteria communicate with each other to generate a cooperative action, bacteria are considered as individuals and social interaction is not used in BC. As a preliminary study, BC concentrated on studying microscopic models that consider the chemotaxis of a single bacterium instead of macroscopic models that analyse the movement of bacteria colonies. Thus for the first time, BC introduced the concept of a 'virtual' bacterium in optimisation algorithms.

In BC, the path of a bacterium is a sequence of straight-line trajectories joined by instantaneous turns, each trajectory being characterised by velocity, direction (angle) and duration.

The velocity v for each individual is assumed to be a scalar constant value:

$$v = \text{const} \tag{3.3.1}$$

The duration of the trajectory γ is selected according to the distribution of a random variable with an exponential probability density function.

$$P(X = \gamma) = \frac{1}{T} \exp^{-\gamma/T} \tag{3.3.2}$$

where T is adaptive according to the fitness of the bacterium.

The new direction is also computed from a probability distribution. The probability density of the angle α between the previous and the new direction follows a Gaussian distribution, with an expectation of 62° and a standard deviation of 26°, as suggested in Berg and Brown's work [6]. Thus the length

of a path l is given by

$$l = v\gamma \tag{3.3.3}$$

.

The normalised new direction vector $\overrightarrow{n_u}$ is multiplied by l to obtain the displacement vector $\overrightarrow{x}$, $\overrightarrow{x} = \overrightarrow{n_u}l$. The new location of the bacterium is

$$\overrightarrow{x}_{\text{new}} = \overrightarrow{x}_{\text{old}} + \overrightarrow{n}_u l \tag{3.3.4}$$

By adopting this strategy, BC is developed to answer the previously unsolved optimisation problems in the following three aspects.

- How to decide the strategy parameters, *i.e.* the minimum duration time T_0, velocity v, *etc.* From an evolutionary perspective, bacteria adapt their motility properties so that they have a better chance to survive in changing environments. However, how parameters are chosen to perform an optimised foraging behaviour needs to be carefully considered.

  In BC, all strategy parameters are optimised by a covariance-matrix adaptation evolution strategy (CMA-ES). According to the chosen target precision ϵ, the initial precision ϵ_{init} and the final precision ϵ_{end}, start the computation searching for the minimum of the function. When the initial precision is reached, the parameters area, such as the chosen strategy parameters T_0, v, are adapted to another precision and the search continues, until this new precision is reached.

- The extension of the 2-D to a multi-dimensional model. Most of the real world problems are high-dimensional rather than 2-D in biological experiments, thus an algorithm which is optimisation-oriented should be considered in a multi-dimensional form.

- The automatic modification of strategy parameters according to the properties of the optimisation function, such as a possibility of escaping

plateaus, and additional features to facilitate the search of global optima. To refine the result, there are also 'mark and explore' strategies to adapt the BC Strategy for the search of Global minima.

It starts from searching the whole domain of the multimodal function with a significantly reduced precision and mark all the points visited by the bacterium. This can be interpreted as the deposition of a marker by the bacterium. The points with the highest frequency of passages are identified by the concentration of this marker and are considered as 'potential' global minima. Then it analyse the concentration of the marker deposited over the whole domain. The neighbourhood of the point with the highest concentration is considered as a candidate for the global minimum. At the second state, it starts a local search with refined parameters from this candidate point and runs until the needed final precision is reached. The final value is chosen as the global minimum of the function reached by the algorithm.

BC has been tested on a number of benchmarks. However, it has been shown that the performance of BC is worse than that of ESs for quadratic functions but comparable for the Rosenbrock function [99]. Therefore, BC has provided a coarse-grained prototype for bacteria-inspired optimisation algorithms and it would be helpful to add the feature of group behaviour into this algorithm.

3.3.2 Bacterial foraging algorithm

Shortly after BC was developed, BFA, which is inspired by the patten exhibited by bacterial foraging behaviour, was proposed [69]. Bacteria have the tendency to gather to nutrient-rich areas by chemotaxis. It is known that bacteria swim by rotating whip-like flagella driven by a reversible motor embedded in the cell wall. *E.coli* has 8-10 flagella placed randomly on the cell body. When all flagella rotate counterclockwise, they form a compact, propelling the cell along a helical trajectory, which is called *Run*. When the flagella rotate clock-

wise, they all pull on the bacterium in different directions, which causes the bacteria to *Tumble*. BFA is based on the tumble and run model.

Chemotaxis

In comparison with BC, bacterial chemotaxis in BFA is solely based on the suppression of tumbles in cells that happen by chance to be moving along an upward gradient. Bacteria make decisions according to their ambient environment. The motion of individual peritrichously flagellated bacteria can be described in terms of run intervals during which the cell swims approximately in a straight line interspersed with tumbles, when the organism undergoes a random reorientation.

In BFA, a unit walk with random direction represents a *Tumble* and a unit walk with the same direction as the last step indicates a *Run*, as shown in Fig. 3.1. After one step move, the position of the i^{th} bacterium can be represented as:

$$\theta_i(j+1,r,l) = \theta_i(j,r,l) + C(i)\angle\phi(j) \tag{3.3.5}$$

where $\theta_i(j,r,l)$ indicates the position of the i^{th} bacterium at the j^{th} chemotactic step in the r^{th} reproductive loop of the l^{th} elimination and dispersion event; $C(i)$ is the length of a unit walk, which is set to be a constant; and $\phi(j)$ is the direction angle of the j^{th} step. When its activity is *Run*, $\phi(j)$ is the same with $\phi(j-1)$; otherwise, $\phi(j)$ is a random angle generated within a range of [0,2π].

With the activity of *run* or *tumble* taken at each step of the chemotaxis process, a step fitness, denoted as $J_i(j,r,l)$, will be evaluated.

Reproduction

The total fitness of each bacterium is calculated as the sum of the step fitness during its life, *i.e.* $\sum_{j=1}^{Nc} J_i(j,r,l)$ which is obtained after all chemotactic steps, where N_c is the maximum number of steps in a chemotaxis process. All bacteria are sorted in reverse order according to their fitness. In the reproduction step, only the first half of population survive and a surviving bacterium

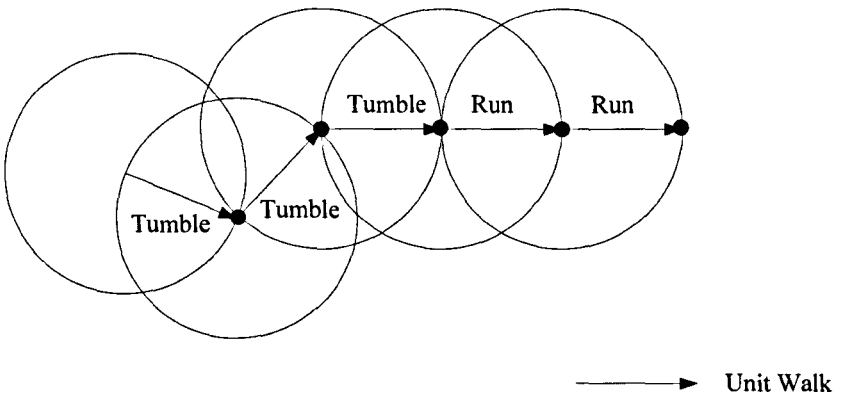


Figure 3.1: Tumble and Run.

splits into two identical ones, which occupy the same positions in the environment at 1st step. Thus, the population of bacteria remains constant in each chemotactic process.

Dispersion and elimination

Chemotaxis provides a basis for local search, and the reproduction process speeds up the convergence as demonstrated by Passino [69]. To a large extent, however, chemotaxis and reproduction are not sufficient for global optima searching. Since bacteria may get stuck around the initial positions or local optima, it is possible for the diversity of BFA to change either gradually or suddenly to eliminate the accidents of being trapped into the local optima. In the BFA, the dispersion event happens after a certain number of reproduction processes. A bacterium is chosen, according to a preset probability p_{ed}, to be dispersed and moved to another position within the environment. These events may prevent the local optima trapping effectively, but unexpectedly disturb the optimisation process. The detailed work can be found in [69].

BFA has made a great contribution to BaIAs. For the first time group behaviour is considered. Together with bacterial chemotaxis, and it yields a satisfying performance in solving low-dimensional optimisation problems. Fig-

ure 3.2 demonstrates the trajectory of bacteria in the first 100 'virtual steps'.

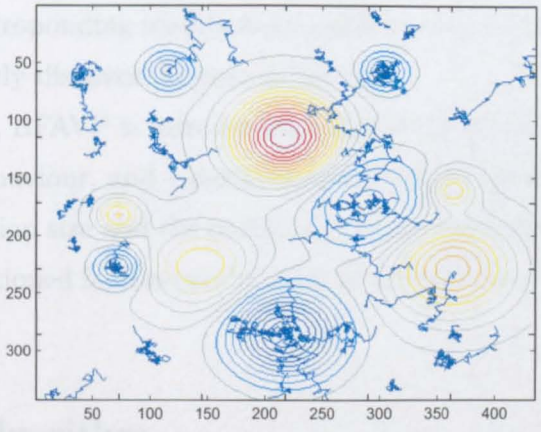


Figure 3.2: Trajectory of bacteria

3.4 Bacterial Foraging Algorithm With Varying Population

The two aforementioned algorithms are preliminary work of BaIAs, which also belong to the EA family. However, like many other existing EAs, they suffer from a number of drawbacks. For instance, BC only explores individual searching capability, *i.e.* bacterial chemotaxis; therefore, a slow convergence rate can be predicted due to the lack of group behaviour. BFA was developed based on a fixed population framework with a coarse-grained 'split and reproduce' mechanism, thus unnecessary computation is inevitably introduced in the optimisation process. To tackle these problems, we need to expand our vision to other related biological phenomena. More characteristics could be analysed and transferred from bacterial foraging into these optimisation algorithms.

As described in Chapter 2, bacteria survive in a large population. However, population size is varying all the time, *i.e.* when nutrient is sufficient, population size will be increasing; whereas when the available nutrient is not enough,

population size could diminish rapidly. In particular, in extreme situations, the whole group can be extinct. This adaptiveness of bacterial population is ensured by its corresponding metabolism, proliferation and elimination system, as well as the newly discovered quorum sensing.

In this section, BFAVP is introduced, which involves further details of bacterial foraging behaviour, and concentrates on describing the relationship between the population size and the complexity of the optimisation problem, and how the aforementioned factors can be organised together to ensure a satisfying performance.

3.4.1 The algorithm

In BFAVP, bacteria follow the chemotactic process, *i.e.* the tumble and run process. This biased random walk also performs 'local search' in problem-solving. Furthermore, in order to reduce the computation load, two properties, namely bacterium's energy and bacterium's age, are incorporated in this algorithm, which assist the simulation of the proliferation and elimination of bacteria. The bacterium's energy is proportional to its fitness value in the environment. Superior bacteria usually contain more energy, which leads to a more filial generation. Based on this feature, bacteria are able to aggregate around optima at an earlier stage of optimisation. Bacterium's age is corresponding to the lifespan of a bacterium. With this mechanism, the computational complexity of the optimisation problem can be reduced and unnecessary computation can be largely avoided, as those bacteria which do not possess sufficient energy during their evolutionary process tend to be eliminated. In order to enhance the group searching performance, a simplified version of quorum sensing is also simulated in this algorithm.

Chemotaxis

Chemotaxis is the tumble-run process that consists of a tumble step and possibly several run steps. Bacteria's positions are represented as 'spots' in an n-dimensional search space, which is also adopted by BC. Suppose the p^{th}

bacterium in the k^{th} chemotactic process, has a current position $\theta_p^k \in \mathbb{R}^n$, and a tumble angle $\varphi_p^k = (\varphi_{p1}^k, \varphi_{p2}^k, ..., \varphi_{p(n-1)}^k) \in \mathbb{R}^{n-1}$. Then the tumble length $D_p^k(\varphi_p^k) = (d_{p1}^k, d_{p2}^k, ..., d_{pn}^k) \in \mathbb{R}^n$ can be calculated from φ_p^k via a Polar-to-Cartesian coordinate transform:

$$\begin{aligned} d_{p1}^k &= \prod_{i=1}^{n-1} \cos\left(\varphi_{pi}^k\right), \\ d_{pj}^k &= \sin\left(\varphi_{p(j-1)}^k\right) \prod_{i=p}^{n-1} \cos\left(\varphi_{pi}^k\right) \quad j = 2, 3, ..., n-1, \\ d_{pn}^k &= \sin\left(\varphi_{p(n-1)}^k\right). \end{aligned} \tag{3.4.1}$$

Note that the maximal tumble angle $\phi_{\max}$ is proportional to the number of dimensions of the objective function, and can be formulated as:

$$\phi_{\max} = \frac{\pi}{(\text{round}(\sqrt{n+1}))^2} \tag{3.4.2}$$

A tumble angle can be generated randomly from the range of $[0, \phi_{\max}]$, which is also the direction followed by the run process, if there is any. Therefore, a step of tumble and run can be expressed as:

$$\varphi_p^k = \varphi_{\text{const}}^k + r_1 \phi_{\max}/2 \tag{3.4.3}$$

$$\hat{\theta}_p^k(1) = X_p^k + r_2 l_{\max} D_p^k(\varphi_p^k) \tag{3.4.4}$$

$$\varphi_{\text{const}}^{k+1} = \varphi_{\text{const}}^k + r_3 \phi_{\max} \tag{3.4.5}$$

where $r_1, r_3 \in \mathbb{R}^{n-1}$ are normally distributed random numbers generated from $\mathcal{N}(0,1)$ and (0,1), respectively, and r_2 is a random number with a range of [-1,1]. $l_{\max}$ the maximal step length of a run, $\hat{\theta}_p^k(1)$ the position of the p^{th} bacterium immediately after the first tumble step, and φ_{const}^k is the angle indicating the searching range for all bacteria in the k^{th} iteration.

Once the angle is set after the tumble step, the bacterium will run for a maximum of n_c steps, or until reaching a position with a worse evaluation value. The position of the p^{th} bacterium is updated at the h^{th} $(h > 1)$ run step in the following way:

$$\hat{\theta}_p^k(h) = \hat{\theta}_p^k(h-1) + r_1 l_{\max} D_p^k(\hat{\varphi}_p^k) \tag{3.4.6}$$

Here, for the convenience of description, the position of the p^{th} bacterium immediately after the tumble-run process of the k^{th} iteration is denoted by $\hat{\theta}_p^k(n_{\text{f}})$, $n_{\text{f}} \leq n_{\text{c}}$.

Metabolism

In VEBAM, the nutrient can be absorbed by bacteria, for chemotactic behaviour and proliferation. Here in BFAVP, energy quantity of the p^{th} bacterium at the k^{th} iteration is denoted by e_p^k. The fitness value of this bacterium at the k^{th} iteration, $J(\theta_p^k)$, can be deemed the source of its energy e_p^k.

In each iteration, a bacterium absorbs energy subsequent to the tumble-run process. The energy transform in the k^{th} iteration is defined as:

$$\tilde{e}_p^k = e_p^k + \xi \cdot J(\theta_p^k) \tag{3.4.7}$$

where ξ is a coefficient for energy transform. For the p^{th} bacterium, there is an upper limit set for proliferation and a lower limit for elimination, which are related to the energy. For instance, if $\tilde{e}_p^k$ reaches the upper limit, the proliferation process is triggered; however, if $\tilde{e}_p^k$ is less than the lower limit, it will be eliminated at the end of the tumble-run process of the $(k+1)^{\text{th}}$ iteration.

Proliferation and elimination

In the proliferation process, bacteria reproduce through binary fission, *i.e.* the cell splits into two identical daughter cells. As aforementioned, the proliferation process is controlled by the bacterium's energy $\tilde{e}_p^k$ in BFAVP. As soon as $\tilde{e}_p^k$ of the p^{th} bacterium reaches the upper limit, it splits into two new individuals. For one of the new bacteria, the bacterium's energy is represented as:

$$\hat{e}_{m+1}^k = \tilde{e}_p^k/2, \tag{3.4.8}$$

where m is the population size in the current iteration, and the other new bacterium keeps the index of its mother cell and also half of the energy as follows:

$$\hat{e}_p^k = \tilde{e}_p^k/2. \tag{3.4.9}$$

For computational purposes, the newly proliferated p^{th} bacterium carries on the computation of metabolism behaviour as its mother cell, *i.e.* $\hat{e}_p^k$ will not change until the next metabolism behavior takes place. However, the new $(m+1)^{\text{th}}$ bacterium will not be involved in the optimisation process until the next tumble-run iteration, thus its energy can be also denoted as e_{m+1}^k.

In order to prevent the population from increasing exponentially, the bacterium's age is also introduced, and works together with the bacterium's energy. For example, for the p^{th} bacterium at the k^{th} iteration, the age is recorded by the counter of η_p^k. In the next iteration, the bacterium's age is changed accordingly:

$$\eta_p^{k+1} = \eta_p^k + 1. \tag{3.4.10}$$

If a cell divides, the ages of the two daughter cells are set to be 0. When a bacterium's age reaches the upper limit of its lifespan, it should be removed from the population. The probability density of lifespan for each individual follows a normal distribution, which reads:

$$\mathcal{P}_p = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(- \frac{(\eta_p^{k+1} - \mu)^2}{2\sigma^2} \right), \tag{3.4.11}$$

where μ is the mean of the bacteria's lifespan, and σ indicates the standard deviation of the bacteria's lifespan. Here, for statistical purposes, the positions of the eliminated cells are recorded.

Quorum sensing

In BFAVP, the quorum sensing modelled in VEBAM is also incorporated. Considering the optimisation problem as a nutrient environment, it is assumed that more 'nutrients' should be located around optima, which correspond to better fitness values. Based on this assumption, the density of the inducer is increased if the bacteria's fitness are generally better. A simplified version of the one modelled in VEBAM is present in BFAVP, *i.e.* most bacteria are attracted by local optima randomly. In this part of computation, updating of

a bacterium's position is described as follows:

$$\theta_p^{k+1} = \delta \cdot (\theta_{\text{best}} - \hat{\theta}_p^k(n_{\text{f}})), \tag{3.4.12}$$

where δ is a coefficient describing the strength of the bacterium's attraction, θ_{best} indicates the position of current best global solution updated after each function evaluation.

However, to avoid premature convergence, a small number of bacteria are randomly selected to be repelled. If the p^{th} bacterium is chosen to enter the repelling process, a random angle in the range of $[0, \pi]$ is generated. The bacterium is thereby 'jumped' with a random step length following this angle in the search space, which can be described as:

$$\theta_p^{k+1} = \hat{\theta}_p^k(n_{\text{f}}) + r_4 l_{\text{range}} D_p^k(\hat{\varphi}_p^k + r_4 \cdot \pi/2), \tag{3.4.13}$$

where $r_4 \in \mathbb{R}^n$ is a normally distributed random sequence drawn from $\mathcal{N}(0, 1)$, and l_{range} is the range of the search space. It should be mentioned that every individual in the population will either be attracted or repelled without exception.

3.4.2 Simulation studies

Simulation Setting

In order to evaluate the performance of BFAVP, thirteen benchmark functions which are divided into three sets are adopted in simulation studies. This work is based on the benchmark functions [39] listed in Appendix A. Functions f_1, f_2, f_4, f_5 and f_6 are high-dimensional uni-modal benchmark functions, used to investigate the convergence rate of each algorithm. Functions $f_8 \sim f_{12}$ are high-dimensional multimodal benchmark functions, which have many local optima, and the total number of local optima increases exponentially as the dimension increases. Hence, it is difficult for most EAs to find the global optimum accurately. Functions $f_{14} \sim f_{16}$ are low-dimensional multi-modal benchmark functions.

BFA, PSO [54], CPSO and EP are compared with BFAVP in the simulation studies. In all experiments, the initial population size of all the algorithms is selected to be 50. For BFAVP, the mean of the bacterium's lifespan, μ, is set to be 50 iterations, *i.e.* 50 tumble-run processes, and the standard deviation, σ, is set to be 10 iterations. N_c is set to be 4. The repelling rate of BFAVP is set to be 0.2, *i.e.* 20 percent of the bacteria will be repelled during the process of quorum sensing. For the parameters of PSO, the inertia weight ω is set as 0.73 and the acceleration factors c_1 and c_2 are both set to 2.05. The CPSO is adopted from [100], which indicates that the use of a constriction factor in the range of $[0, 1]$ may be necessary to ensure convergence of the CPSO. The tournament size is set to be 10 for selection in EP .

3.4.3 Simulation results and discussion

As it can be seen from Table 3.1, CPSO outperforms other EAs for most of the uni-modal functions. Nonetheless, BFAVP obtains a stabler performance than BFA, PSO and EP for most of the functions. Figure 3.3 demonstrates the convergence process of these five algorithms in solving the Sphere's problem.

The results listed in Tables 3.2 and 3.3 are obtained from the evaluation on multimodal benchmark functions for high-dimensional functions and low-dimensional ones respectively. It is shown that multimodal problems can be solved effectively by BFAVP, due to its varying population scheme, which enables it to achieve better performance with the same number of function evaluation. Figures 3.4 and 3.5 illustrate the performance of the listed five algorithms in Schwefel's function and Rastrigin's function, respectively. Schwefel's function is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction. It can be observed in Fig. 3.4 that there is a big gap between the best result obtained by BFAVP and the next best algorithm, CPSO; the other three algorithms perform even worse. Rastrigin's function is based on Sphere's function with the addition of cosine modulation to produce many local minima. However,

Table 3.1: Average Best Results and Standard Deviations of BFAVP, BFA, PSO, CPSO and EP for uni-modal functions

Function	BFAVP	BFA	PSO	CPSO	EP
f_1	1.9848×10^{-5}	2.8743×10^{-2}	11.7381	7.3170×10^{-61}	2.0800×10^{-22}
	(1.0894×10^{-5})	(28540×10^{-3})	(6.8628)	(1.6407×10^{-60})	(8.1789×10^{-24})
f_2	1.0648×10^{-3}	8.6442	0.7664	1.5541×10^{-15}	6.26572×10^{-12}
	(7.8569×10^{-4})	(17.1490)	(0.2219)	(8.4929×10^{-15})	(1.5593×10^{-13})
f_4	50.8701	3.8078	501.7887	3.1957×10^{-6}	69.5189
	(31.9344)	(12.7994)	(135.3998)	(4.9756×10^{-6})	(45.8909)
f_5	0.3297	12.8941	4.8835	2.3865×10^{-4}	6.4993×10^{-12}
	(0.1458)	(4.1394)	(0.6045)	(3.4761×10^{-4})	(1.9971×10^{-13})
f_6	20.4322	4.0946×10^{2}	8.7329×10^{4}	39.6272	2.6392×10^{2}
	(1.0978×10^{2})	(2.2369×10^{3})	(1.0227×10^{5})	(1.5080×10^{2})	(9.0153×10^{2})

Table 3.2: Average Best Results and Standard Deviations of BFAVP, BFA, PSO, CPSO and EP for high-dimensional multi-modal functions

Function	BFAVP	BFA	PSO	CPSO	EP
f_8	-12213.0879	-7126.0912	-7097.5321	-6664.3791	-9080.5037
	(338.8019)	(648.3530)	(829.2348)	(874.9836)	(660.4163)
f_9	9.2972	72.9443	50.4906	46.6635	18.9373
	(3.0271)	(17.6868)	(14.37599)	(13.9913)	(4.8354)
f_{10}	1.4193×10^{-3}	14.6715	2.0143	0.8248	0.1391
	(1.4064×10^{-3})	(0.8448)	(0.5417)	(0.7631)	(0.3631)
f_{11}	1.3920×10^{-2}	0.1021	1.4570	2.3868×10^{-2}	1.8292×10^{-2}
	(3.0061×10^{-2})	(6.7710×10^{-2})	(0.1779)	(3.5925×10^{-2})	(3.5037×10^{-2})
f_{12}	4.7498×10^{-5}	26.0503	1.0565×10^{4}	0.2110	0.3853
	(1.1948×10^{4})	(8.9542)	(1.0341×10^{4})	(0.2993)	(0.5189)

Table 3.3: Average Best Results and Standard Deviations of BFAVP, BFA, PSO, CPSO and EP for low-dimensional multi-modal functions

Function	BFAVP	BFA	PSO	CPSO	EP
f_{14}	0.9980	12.1168	2.6065	3.0638	1.0642
	(4.7612×10^{-14})	(6.2813)	(2.5707)	(2.9457)	(0.2521)
f_{15}	5.9637×10^{-4}	1.7424×10^{-2}	1.3739×10^{-3}	1.0358×10^{-3}	8.9079×10^{-4}
	(2.7141×10^{-4})	(2.2276×10^{-2})	(3.5982×10^{-3})	(3.6554×10^{-3})	(8.5889×10^{-4})
f_{16}	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	(4.8085×10^{-16})	(8.3205×10^{-9})	(5.5199×10^{-15})	(6.2532×10^{-16})	(1.0792×10^{-4})

the location of the minima are regularly distributed. It is shown in Fig. 3.5 that BFAVP is able to search for better local minima in the function evaluation process, in comparison with the premature convergence of the other four algorithms. Figure 3.6 also demonstrates the similar characteristic of BFAVP in solving low-dimensional problems.

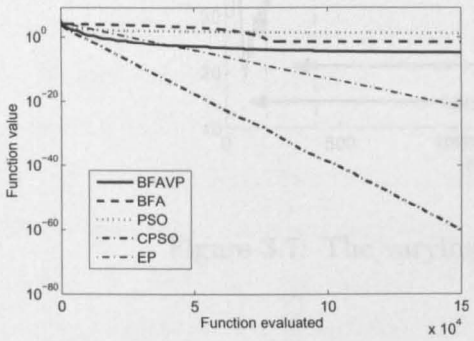


Figure 3.3: Convergence process for f_1

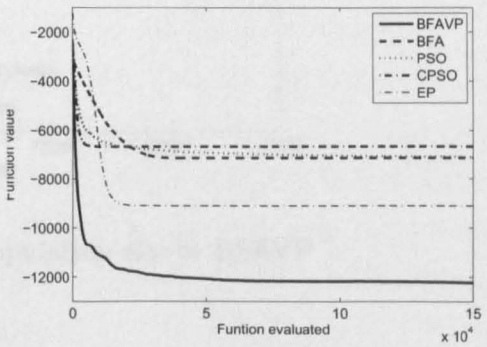


Figure 3.4: Convergence process for f_8

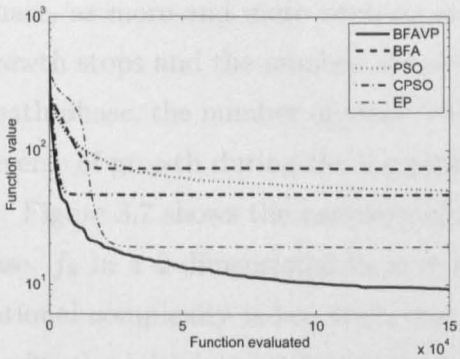


Figure 3.5: Convergence process for f_9

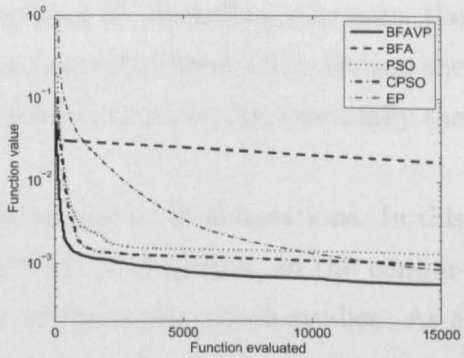


Figure 3.6: Convergence process for f_{12}

3.4.4 Variation of population size

It is reported from theoretical studies that bacterial growth follows four phases: lag phase, log phase, stationary (stable) phase and death phase [90]. Lag phase happens immediately after the introduction of the cells into the

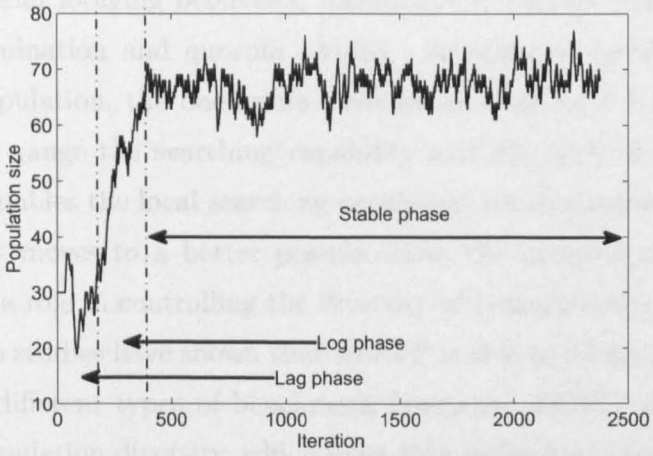


Figure 3.7: The varying population size in BFAVP

nutrient environment, and the population remains temporarily unchanged. As soon as cells are dividing regularly by binary fission, it enters into the exponential phase, *i.e.* the doubling time of the bacterial population. In the stationary phase, as more and more bacteria are competing for dwindling nutrients, the growth stops and the number of bacteria becomes stabilised [101]. During the death phase, the number of viable cells decreases exponentially, essentially the reverse of growth during the log phase.

Figure 3.7 shows the variation of population size in 2500 iterations. In this case, f_8 in a 2-dimensional form is adopted for investigation, so the computational complexity is less than that of the previous simulation studies. As a result, the initial population is reduced from 50 to 20. The mean maximum population limitation is set to be 150. It can be observed from Fig. 3.7 that the lag phase, the log phase, and the stable phase exhibited by the variation of population are roughly in line with their counterparts in biological studies.

3.4.5 Discussion

An effective optimisation algorithm, BFAVP, is proposed to bridge the gap between modelling and optimisation. It is inspired by a number of character-

istics of bacterial foraging behaviors, including chemotaxis, metabolism, proliferation, elimination and quorum sensing. In order to build a framework of varying population, the bacterium's energy and age have been introduced to BFAVP to gauge the searching capability and life cycle of an individual. Chemotaxis enables the local searching capability, which ensures that the bacterium always moves to a better position than the previous step. Quorum sensing plays a role in controlling the diversity of bacterial population.

Simulation studies have shown that BFAVP is able to adapt the population according to different types of benchmark functions. BFAVP also overcomes the lack of population diversity, which most EAs suffer from, and it has better computation efficiency than other EAs. With the flexible operation in quorum sensing, BFAVP is more suitable for high-dimensional multimodal functions than the other EAs adopted for comparison.

3.5 Bacterial Swarming Algorithm

Over the past few decades, many biologically inspired computational methodologies have been invented. However, the running of EAs may be time consuming in searching along the directions which are randomly selected. Therefore they have slow convergence rates and are reluctantly used in many large optimisation problems. Although hybrid EAs have been introduced [102], these hybrid EAs require the objective function to be differentiable. Recently, two SI paradigms have been developed to tackle many optimisation problems for which robust solutions are difficult or impossible to find using traditional approaches. One of them is ACO [61], which is inspired by ant routing behavior. The other is PSO [54], simulating animal swarm behavior. However, although the EAs and SI have been comprehensively studied, these methods face difficulties in applications to large-scale high-dimensional optimisation problems, primarily because of the huge computational burden they impose. A key advance in this field will therefore be made by a significant reduction in the computational time-costs whilst further improving the efficiency of global search capabilities

of these algorithms.

More recently, the study of BFA [69] has received great attention in the Computational Intelligence community worldwide. BFA is based on study of the bacterial foraging behaviors. The complex and organised activities exhibited in bacterial foraging patterns could inspire a new solution for optimisation problems. The underlying mechanism of the surviving of bacteria, especially *E.coli*, in a complex environment has been reported by researchers in the field of biological sciences [6]. Inspired by these phenomena, BFA was developed as an optimisation algorithm, in which the self-adaptability of individuals in the group searching activities has attracted a great deal of interest. However, in BFA, the chemotactic process is artificially set, hence this process could not achieve convincing results in a certain range of optimisation problems, especially high dimensional and multi-modal problems.

As a consequence, we propose a novel optimisation algorithm, BSA. BSA is based on BFA but incorporates ideas from the modelling of bacterial foraging patterns studied recently [103]. The behaviour of tumble and run actions of bacteria and modelling of these actions have been further incorporated to develop BSA. To improve the performance of BSA for complex optimisation problems, we have also introduced an adaptive step length of run actions, which is independent of the optimisation problems but can speed up the convergence process of BSA. To demonstrate the merits of BSA, we have evaluated it on a number of mathematical benchmark functions which cover a range of optimisation problems from uni-modal and multi-modal to low and high dimensions. The algorithm evaluation has been undertaken in comparison with FEP and PSO. Finally, the simulation results are provided to support our discussions on BSA.

3.5.1 Mathematical framework and algorithm

The role of cell-cell communication

Cell-cell communication is concerned with the ability of exploration and exploitation in an optimisation algorithm. Communication between bacteria, by the mechanism referred to 'quorum sensing', is widespread in nature. This phenomenon is related to the density of a colony. The higher the density, the more attraction would be generated by a colony to the individuals outside of the colony, which will influence bacterial foraging patterns and development of colonies during the whole evolutionary process. This bacterial phenomenon is similar to that emulated in PSO. PSO adopts the attraction of the particles as the major or determining factor of the algorithm. Each particle in PSO updates its position at each iteration according to its historical best position and the global best position in that iteration. The direction from the current position to the global best position can be regarded as the communication between particles, while the other one towards the historical best can be regarded as random walk. The inertia weight vector used in PSO can effectively stop the particles converging prematurely.

In BSA, the position of the global best is supposed to be the center of a colony, and a simplified cell-cell communication is adopted, in a similar way to PSO. Apart from the attraction, we also introduce dispersion events into BSA. The cells are dispersed to the area near the best performed ones according to a preset probability. As a result, the combination of attraction and dispersion events ensures the diversity and the searching ability of this algorithm.

Mathematical representation

Bacterial chemotaxis is based on the suppression of tumbles in cells that happen by chance to be moving up along an upward gradient. Bacteria make decision according to their ambient environment. The motion of individual peritrichously flagellated bacteria can be described in terms of run intervals during which the cell swims approximately in a straight line interspersed with

tumbles, when the organism undergoes a random reorientation.

In BSA, a unit walk with random direction represents a *Tumble* and a unit walk with the same direction of the last step indicates a *Run*. Similar to BFA, a chemotactic process in BSA consists of one step of *Tumble* and N_s steps of *Run*, depending on the variation of the environment.

In the process of *Tumble*, the position of the i^{th} bacterium can be represented as

$$\theta_i(j+1,r) = \theta_i(j,r) + C_i(j)\angle\phi_i(j) \tag{3.5.1}$$

where $\theta_i(j,r)$ indicates the position of the i^{th} bacterium at the j^{th} chemotactic step in the r^{th} iteration loop; $C_i(j)$ is the length vector of a unit walk for the i^{th} bacterium at the j^{th} chemotactic step and $\angle\phi_i(j)$ is the direction angle of the j^{th} step for bacterium i, and it is a random angle generated within the range $[0,\ 2\pi]$.

However in BSA, in order to accelerate the convergence rate and enhance its searching ability in different types of problems. $C_i(j)$ is set to be adaptive and is defined as follows:

$$C_i(j) = \begin{cases} C_{\text{init}} \times B, & : \text{ if Tumble} \\ D_1 \times r_1 \times B, & : \text{ if Run} \end{cases} \tag{3.5.2}$$

where C_{init} is the step size for a unit walk, D_1 a constant, r_1 a random number, and $r_1 \in [0,1]$, B is the length vector of the boundaries of the search domain.

The fitness of the i^{th} bacterium at the j^{th} chemotactic step is represented by $J_i(j,r)$. If $J_i(j+1,r)$ is better than $J_i(j,r)$, then the process of *Run* follows, which can be represented by:

$$\hat{\theta}_i^{k+1}(j+1,r) = \hat{\theta}_i^k(j+1,r) + C_i(j)\angle\phi_i(j) \tag{3.5.3}$$

where $1 \le k \le N_s$, and $\hat{\theta}_i^k$ denotes the position of the i^{th} bacterium in the k^{th} step of *Run*. This process continues until $J_i^{k+1}(j+1,r)$ is worse than $J_i^k(j+1,r)$. The angle $\angle\phi_i(j)$ remains unchanged during this process.

After each chemotactic process, the energy obtained by a single bacterium in its life time, which is denoted by N_c, is accumulated. Compared with the others, the individual with most energy gained is defined as the best cell and its position $\theta^p(r)$ is kept for the calculation of the other bacteria in the next selection process. A certain percentage of bacteria undergo an attraction action, which are selected with probability p_a. Based on both their current positions and the global best position, their positions are recalculated as follows.

$$\theta_i(1, r+1) = \theta_i(N_c, r) + r_2 \times D_2 \times (\theta^p(r) - \theta_i(N_c, r)) \tag{3.5.4}$$

where D_2 is a constant and r_2 is a random number, $r_2 \in [0, 1]$. It should be mentioned that in the above equation the position of each bacterium starts to be updated by the attraction action using its position at the last step of the current chemotactic process for the first step of the next chemotactic process. This is why the index of chemotactic process is set to be 1 on the left side of the above equation.

Figure 3.8 illustrates the mechanism for attraction and dispersion. This simplified cell-cell communication process shares some similarities with PSO while keeping its simplicity by only using the global best, which reduces the computational complexity to a certain extent. The rest of the bacteria are dispersed to positions around the best individual with a randomly chosen mutation step and mutation angle, using the following equation:

$$\theta_i(1, r+1) = \theta^p(r) + r_3 B \angle \psi \tag{3.5.5}$$

where r_3 is a random number, $r_3 \in [0, 1]$, and ψ is a random angle chosen from $[0, 2\pi]$.

Pseudo code

The pseudo code of BSA is given in Table 3.4, which indicates the executable procedure of the mathematical framework and shows the implementation of BSA.

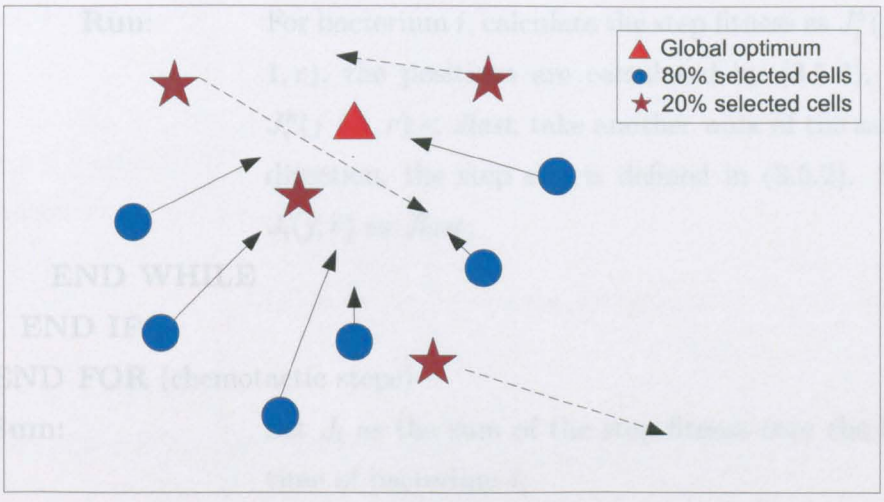


Figure 3.8: Attraction and dispersion

Table 3.4: Pseudo code of BSA

Randomly initialize positions of bacteria in the domain;

FOR (Selection $r = 1 : N_s$)

FOR (chemotactic steps per bacterium $j = 1 : N_c$)

Calculate: Calculate the nutrient function of bacterium i as $J_i(j, r)$;

Tumble: For bacterium i, set $J_i(j, r)$ as *Jlast*.

Generate a random angle represented by an array X, where each element belongs to $[0, 1]$;

Move to a random direction $\frac{X}{\|X\|}$ by a unit walk by step size C_{init}, the new position is calculated by (3.5.1); Start another chemotactic step.

IF $\left(J_i(j+1, r) < J_i(j, r)\right)$

WHILE $(J_i^{k+1}(j+1, r) < J_i^k(j+1, r)$ and $k < N_c)$

Run: For bacterium i, calculate the step fitness as $J_i^k(j+1,r)$, the positions are calculated by (3.5.3). If $J_i^k(j+1,r) < Jlast$, take another walk of the same direction, the step size is defined in (3.5.2). Set $J_i(j,r)$ as $Jlast$;

END WHILE

END IF

END FOR (chemotactic steps)

Sum: Set J_i as the sum of the step fitness over the life time of bacterium i;

Sort: Sort J_i in ascending values of fitness in the population;

Communication : Calculate the rank of bacterium i according to their fitness in its life time; keep the position of the fittest individual; randomly select the rest of the individuals, by a probability p_a, to undergo attraction using (3.5.4); other individuals undergo dispersion using (3.5.5).

END FOR (Selection)

3.5.2 Simulation studies

The benchmark functions

In order to evaluate the performance of BSA, seven benchmark functions selected from [39] are listed below.

Generalized Rosenbrock's Function:

$$f_1(x) = \sum_{i=1}^{29}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{3.5.6}$$

where $x \in [-30, 30]^{30}$;

Step Function:

$$f_2(x) = \sum_{i=1}^{30}(\lfloor x_i + 0.5 \rfloor)^2 \tag{3.5.7}$$

where $x \in [-100, 100]^{30}$;

Quartic Function, *i.e.*, Noise:

$$f_3(x) = \sum_{i=1}^{30} i x_i^4 + random[0, 1) \tag{3.5.8}$$

where $x \in [-1.28, 1.28]^{30}$;

Generalized Schwefel's Problem 2.26:

$$f_4(x) = -\sum_{i=1}^{30}(x_i \sin(\sqrt{|x_i|})) \tag{3.5.9}$$

where $x \in [-500, 500]^{30}$;

Generalized Rastrigin's Function:

$$f_5(x) = \sum_{i=1}^{30} x_i^2 - 10\cos(2\pi x_i) + 10 \tag{3.5.10}$$

where $x \in [-5.12, 5.12]^{30}$;

Ackley's Function:

$$\begin{aligned} f_6(x) \;=\; & -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) \\ & -\exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right) + 20 + e \end{aligned} \tag{3.5.11}$$

where $x \in [-32, 32]^{30}$;

Shekel's Foxholes Function:

$$f_7(x) = \left[\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1} \tag{3.5.12}$$

where $x \in [-65.536, 65.536]^2$, and

$$a = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$$

The benchmark functions $f_1 \sim f_6$ are well-known. They are high dimensional, and among them $f_1 \sim f_3$ are uni-modal functions, $f_4 \sim f_6$ are multi-modal functions. The function f_7 falls into the category of low dimension functions.

Parameter settings

Table 3.5: The evaluation numbers

Function	FEP	PSO	BSA
f_1	2,000,000	200,000	200,000
f_2	150,000	200,000	200,000
f_3	300,000	200,000	200,000
f_4	900,000	200,000	200,000
f_5	500,000	200,000	200,000
f_6	150,000	200,000	200,000
f_7	10,000	200,000	5,000

In the simulation studies, BSA is evaluated on the benchmark functions in comparison with FEP and PSO. Here, the number of evaluations of an objective function in each algorithm is adopted for comparison purpose. The total evaluation number for each algorithm, taken in a complete optimisation process, is listed in Table 3.5. The evaluation numbers of FEP and PSO are used in [39] and [104], respectively. The parameters of FEP and PSO are set as indicated in [39] and [104], respectively. The population of each algorithm is set as 50. The cross over rate for FEP is set to be 0.7. The parameters in BSA are set as $C_{\text{init}} = 0.001$, $D_1 = 0.5$, $D_2 = 1$ and $p_a = 0.8$.

Simulation results

BSA, FEP and PSO are used to optimise the benchmark functions respectively. Each algorithm was run 20 times to give a mean value of the best solutions and a standard deviation obtained from the 20 runs. Table 3.6 shows the results obtained by the three algorithms applied on the seven benchmark

Table 3.6: Simulation results

Function	FEP		PSO		BSA	
	Mean Best	Std Dev	Mean Best	Std Dev	Mean Best	Std Dev
f_1	32.5321	18.7004	30.1393	25.8183	2.4657	3.0588
f_2	0	0	0.1	0.3162	0.15	0.6708
f_3	7.4×10^{-4}	2.319×10^{-4}	2.5×10^{-3}	5.28×10^{-4}	8.7203×10^{-4}	7.5285×10^{-3}
f_4	-11397	364	-7467	980.7	-12567.5	3.25
f_5	13.92	3.9577	56.5371	19.6147	1.5051	6.6717
f_6	0.4482	0.3869	2.32×10^{-9}	1.33×10^{-9}	6.87×10^{-2}	7.78×10^{-2}
f_7	1.1968	0.4192	1.3952	0.6940	0.9980	0

functions. In this table, the results of FEP and PSO were generated using the programs developed by ourselves and they are close to the results reported in [39]. It should be mentioned that the result of f_5 is significantly different from that given in [39], although we have attempted to achieve the same result without success. This will remain as an issue for us to investigate in the near future. From Table 3.6, it can be seen clearly that BSA can provide a better optimisation solution with a much smaller deviation for four out of seven benchmark functions, encompassing both uni-modal and multi-modal problems with low and high dimensions. The merits and characteristics of BSA are discussed in comparison with FEP and PSO as follows.

Convergence

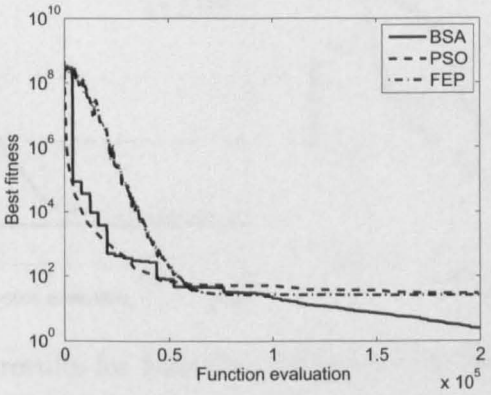


Figure 3.9: Best results for Function f_1

Figures 3.9 $\sim$ 3.15 show the convergence processes of BSA, FEP and PSO respectively, conducted on the seven benchmark functions. The three algorithms take 200,000 function evaluations for functions $f_1 \sim f_6$ and 5,000 function evaluations for function f_7. It should be mentioned here that a generation of BSA involves more evaluations of the objective function than that taken by FEP and PSO, since the generation is concerned with the life time of a bacterium, which consists of an unfixed number of chemotactic processes. For comparison purposes, we use the number of evaluations to plot the convergence

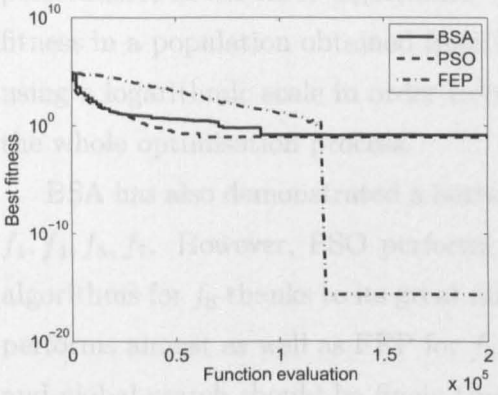


Figure 3.10: Best results for Function f_2

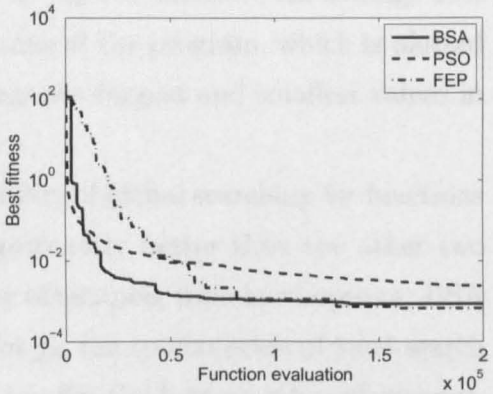


Figure 3.11: Best results for Function f_3

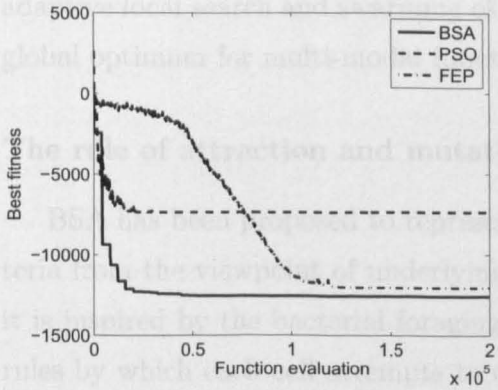


Figure 3.12: Best results for Function f_4

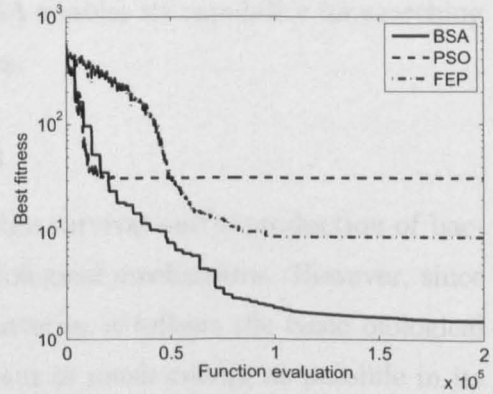


Figure 3.13: Best results for Function f_5

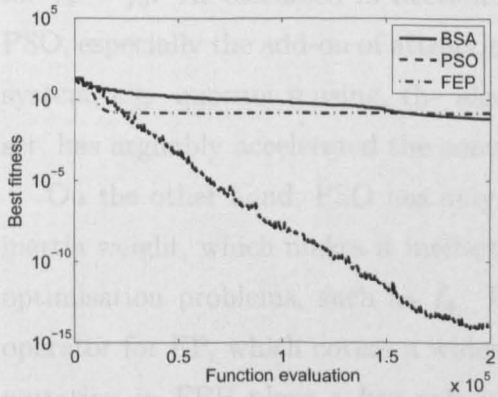


Figure 3.14: Best results for Function f_6

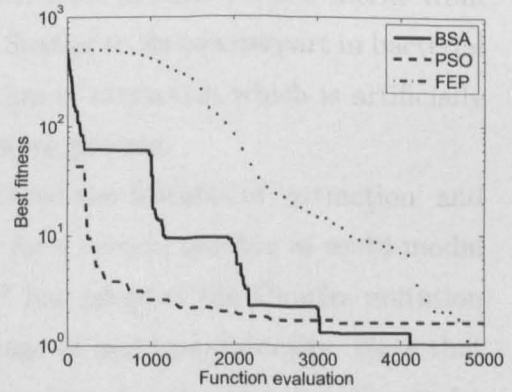


Figure 3.15: Best results for Function f_7

performance of the three algorithms. All the figures illustrate the average best fitness in a population obtained from 20 runs of the program, which is plotted using a logarithmic scale in order to reduce the biggest and smallest values in the whole optimisation process.

BSA has also demonstrated a better ability of global searching for functions f_1, f_4, f_5, f_7. However, PSO performs significantly better than the other two algorithms for f_6 thanks to its great ability of escaping from local optima. BSA performs almost as well as FEP for f_3. For f_3, the combination of local search and global search should be finely tuned to offer the best search performance. From the results presented in Table 3.6, it can be seen that BSA performs the best on average for most of the benchmark functions. This is because the adaptive local search and swarming of BSA enables its capability for searching global optimum for multi-modal functions.

The role of attraction and mutation

BSA has been proposed to represent the survival and reproduction of bacteria from the viewpoint of underlying biological mechanisms. However, since it is inspired by the bacterial foraging patterns, it follows the basic biological rules by which each cell attempts to obtain as much energy as possible in its lifetime. The simulation results show that BSA usually converges to the global optimum at an early stage for most of the evaluation functions, and specifically for $f_4 \sim f_5$. As discussed in Section 3.5.1, BSA inherits certain merits from PSO, especially the add-on of attraction. Similar to its counterpart in bacterial system, e.g. quorum sensing, the algorithm of attraction which is artificially set, has arguably accelerated the convergence process.

On the other hand, PSO has only utilised the features of 'attraction' and inertia weight, which makes it ineffective for a certain number of multi-modal optimisation problems, such as f_4. FEP has adopted the Cauchy mutation operator for EP, which covers a wider range of mutation intensity. Note that mutation in FEP plays a key role in searching for the global optimum for f_2. In this step function, the range of this mutation increases exponentially

as the dimension gets higher. As f_2 is evaluated with a higher dimension, therefore, a greater range of mutation enables FEP to be more capable of global search. FEP performs very well for this benchmark function while the other two algorithms focus more on local searches. Thus they do not have a sufficient level of mutation, which hampers their performance for this function.

Robustness

In most EAs, the algorithm robustness is a crucial issue, as EAs are based on stochastic research and random selections. This issue is also considered in PSO. Regarding bacterial foraging behaviors, by an appropriate observation of population development and environmental difference in a bacterial foraging process, it has been well noted that bacteria have a potential capability of growing in and moving towards rich nutrient areas. The behavior is more certain than that emulated in most of EAs and PSO. Although the sensitivity to the initial positions and the instability in different functions have been noted during the course of the simulation studies, the simulation results show that the standard deviations obtained by BSA for most of the benchmark functions are smaller than those obtained by FEP and PSO.

3.5.3 Discussion

The novel BSA has been proposed for global optimisation. In this algorithm, the adaptive tumble and run operators have been developed and incorporated, which are based on the understanding of the details of the bacterial chemotactic process. The operators involve two parts: the first is concerned with the selections of tumble and run actions, based on their probabilities which are updated during the searching process; the second is related to the length of run steps, which is made adaptive and independent of the pre-acquired knowledge of optimisation problems. These two parts are utilised to balance the global and local searching capabilities of BSA. Beyond the tumble and run operators, attraction and mutation operations have also been developed. The former is inspired from the quorum sensing phenomenon of bacterial foraging

patterns, incorporating the idea of PSO attraction, and the latter, considering the dispersion events of the growing process of bacteria, plays a significant role in keeping a certain diversity of bacterial population for retaining a global searching capability, in particular in the cases of optimisation of high dimensional functions.

BSA has been evaluated on a number of benchmark problems, which include uni-modal and multi-modal functions in low and high dimension domains, respectively. The convergence rates and robustness of BSA have been discussed in this section. The simulation results have shown that BSA has a superior performance in comparison with FEP and PSO. Through the simulation studies, it has been seen that BSA possesses a great potential for global optimisation of complex problems.

3.6 Paired-bacteria Optimiser

EAs are notorious for their intensive computation caused by a large number of evaluations of the objective function required by all individuals in every single searching process. Most of the EAs introduce unnecessary computation in calculating for the badly performing individuals. This is caused by the nature of uncertainty in association with the random search undertaken in the optimisation process. It should be noted that the higher dimension the objective function is, the more uncertainties an EA has to face. For example, in an n-dimension search space, an individual would have n directions to move or move in an arbitrary direction which is in fact composed by n coordinates. Therefore for the computation purpose, n random numbers have to be generated for n coordinates respectively to form this arbitrary position of the individual. Apparently, the uncertainty of the search process can be reduced if an individual is forced to move on a coordinate direction or a few coordinate directions, and consequently the amount of unnecessary computation can be significantly reduced. It may be argued that forcing an individual to move on a single coordinate would weaken the global search capability. Therefore, the coordinate,

on which the position of the individual changes, should be randomly selected at each search step.

Based on this idea, thus, instead of adopting the commonly used population based computation, we propose a PBO, which has only a pair of individuals in a population, the primary individual and the pseudo individual. The primary individual play a role in gradient-based search and the pseudo individual functions a random walk in an arbitrary coordinate to keep the nature of random search. The primary individual can also be regarded as a bacterium which evolves in a sufficient rich nutrient area. The random bacterium can always be seen as an explorer. It also incorporates certain features of other widely used EAs, such as PSO, to enhance the cooperation between these two bacteria. The simulation results illustrate its advantage when applied to multi-modal problems. The performance in multi-modal problems is intensively discussed and compared with the results obtained by BFA and FEP. The potentials of parallel computation for this algorithm are also discussed.

3.6.1 The algorithm of PBO

Only two individuals are involved in the computation, a primary bacterium, X and a pseudo one, $\tilde{X}$. The primary bacterium, X, performs basic search in the problem space. The position of X at the k^{th} iteration, θ, is defined as:

$$\theta^k = [\theta_1^k, \theta_2^k, ..., \theta_n^k] \tag{3.6.1}$$

where θ_i^k is the i^{th} component of θ^k and n the number of dimensions.

Meanwhile, the pseudo bacterium $\tilde{X}$ accompanies X at each iteration to function as a mutation of bacterium X. Choose a random dimension l, $l \in \{1, 2, \cdots, n\}$, the position of the pseudo bacterium $\tilde{\theta}$ at the k^{th} iteration is placed as:

$$\tilde{\theta}^k = \theta^k + \Delta D^k \tag{3.6.2}$$

where $\Delta D^k = [0, \cdots, 0, d_l^k, 0, \cdots, 0]$, and d_l^k indicates a value randomly chosen

between the lower boundary and the upper boundary of the search space on the l^{th} dimension. Furthermore, d_l^k is chosen according to a parameter m, which defines the proportion of the distance to the boundary.

According to the fundamental theory of bacterial chemotaxis, environmental change can be represented roughly by the gradient change. Set g_l^k as an alternative format of pseudo gradient [105] along the l^{th} dimension at the k^{th} iteration, as follows.

$$g_l^k = \frac{f(\tilde{\theta}^k) - f(\theta^k)}{\tilde{\theta}_l^k - \theta_l^k} \tag{3.6.3}$$

where $f(\theta^k)$ and $f(\tilde{\theta}^k)$ are the fitness of θ^k and $\tilde{\theta}^k$, respectively.

The primary bacterium has its own velocity, denoted by C^k, $C^k = [c_1^k, c_2^k, ..., c_n^k]$, where c_i^k is the i^{th} component of C^k at the k^{th} iteration. The gradient g_l^k is added to the l^{th} component of velocity as follows:

$$C^k = [c_1^k, c_2^k, \cdots, c_l^k + g_l^k, \cdots, c_n^k] \tag{3.6.4}$$

The velocity is updated at the k^{th} iteration as follows:

$$C^{k+1} = \overrightarrow{\omega}^k * (C^k + n_1 * r * (P_{\text{p}}^k - C^k)) \tag{3.6.5}$$

where $\overrightarrow{\omega}^k = [\omega_1^k, \cdots, \omega_n^k]$ indicates the inertia weight vector of the bacterium velocity C^k; P_{p}^k the position of the best bacterium in the past k iterations; n the attractor factor, which is a constant; and r a random number, $r \in [0, 1]$. C_i^{k+1} is clipped into the range of $[0, C_i^{\max}]$, where $C_i^{\max}$ is the maximum velocity of bacterium X along the i^{th} dimension, which is scaled proportionally to the maximum value of search boundary of that dimension.

Hence, the position of the primary bacterium is updated at the k^{th} iteration as:

$$\theta^{k+1} = \theta^k + C^k \tag{3.6.6}$$

This process is continually performed until a termination criterion is reached. The overall operation of PBO is as follows:

Table 3.7: Pseudo code of PBO

Initiate the positions of the primary bacterium and pseudo bacterium;	
Initiate the velocity of the primary bacterium;	
Evaluate the fitness of the primary bacterium;	
Set G_{p}^{0} as the value of global optimum;	
WHILE (the termination conditions are not met)	
Select a dimension l randomly;	
Pseudo bacterium:	Move $\tilde{X}$ along dimension l by (3.6.2); Evaluate fitness of $\tilde{X}^{k}$;
Position updating:	Calculate the pseudo gradient along the l^{th} dimension using (3.6.3); Update the step length of primary bacterium along dimension l by (3.6.4); Calculate the step length of primary bacterium by (3.6.5);
Generation:	Update the position of primary bacterium by (3.6.6); Update G_{p}^{k} if the fitness of current bacterium is better;
$k = k + 1$;	
END WHILE	

3.6.2 Simulation studies

PBO is evaluated in comparison with PSO [54] and FEP [39], which are widely used to test an EAs' efficiency [39]. The evaluation is undertaken on four benchmark minimization problems given as follows:

$$
\begin{aligned}
\text{F1}: f_1(X) &= \sum_{i=1}^{30} x_i^2 \\
\text{F2}: f_2(X) &= -\sum_{i=1}^{30} (x_i \sin(\sqrt{|x_i|})) \\
\text{F3}: f_3(X) &= \sum_{i=1}^{30} (x_i^2 - 10\cos(2\pi x_i) + 10)^2 \\
\text{F4}: f_4(X) &= -20\exp\left(-0.2\sqrt{\tfrac{1}{30}\textstyle\sum_{i=1}^{30} x_i^2}\right) \\
&\quad - \exp\left(\tfrac{1}{30}\textstyle\sum_{i=1}^{30} \cos 2\pi x_i\right) + 20 + e
\end{aligned}
$$

Table 3.8: Best fitness and evaluation times

Functions	Algorithms	Evaluations	Mean	Std Dev
F1	PSO	150,000	1.4638×10^{-20}	5.2324×10^{-20}
	FEP	150,000	2.5842×10^{-14}	5.7785×10^{-14}
	PBO	100,000	7.3948×10^{-4}	9.2655×10^{-4}
F2	PSO	150,000	−9717.1033	714.7094
	FEP	150,000	−11397	364
	PBO	100,000	−12569.4603	0.01954
F3	PSO	150,000	56.5377	19.6174
	FEP	150,000	13.92	3.9577
	PBO	100,000	0.1584	0.4176
F4	PSO	150,000	0.3260	0.6127
	FEP	150,000	0.4482	0.3869
	PBO	100,000	8.1896×10^{-3}	3.2052×10^{-3}

The parameters are set as follows: m is chosen as 0.05 or 0.1, randomly. n is set as 7 by trial and $\overrightarrow{\omega}$ is set as 0.3 in each dimension. $C_i^{\max}$ is set as 0.1% of the maximum length along the i^{th} dimension.

These objective functions are evaluated respectively by PSO and FEP for 150,000 times while they are evaluated for 100,000 times by PBO. The average

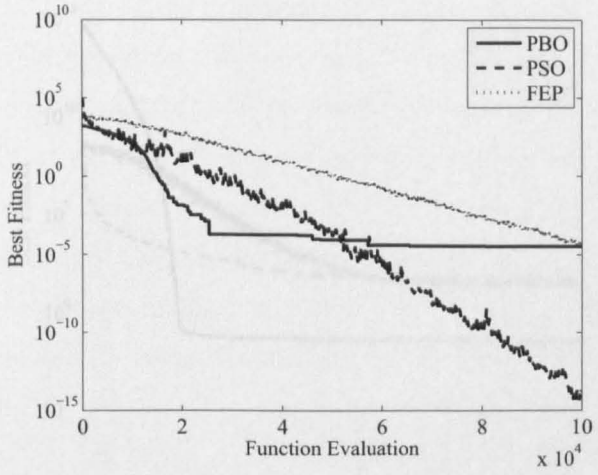


Figure 3.16: The convergence process for f_1

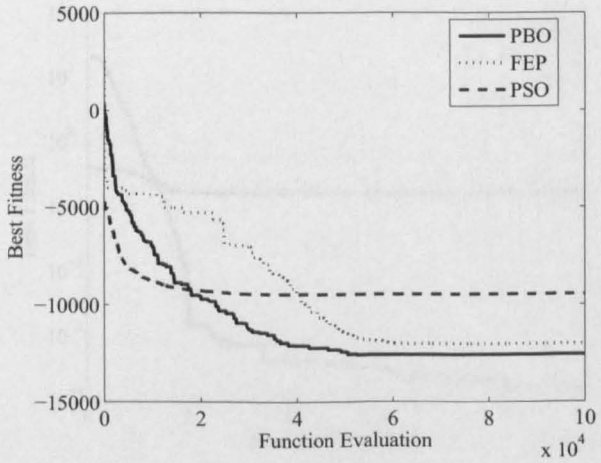


Figure 3.17: The convergence process for f_2

best results obtained in 50 runs are listed in Table 3.8. From Table 3.8, it can be seen clearly that PBO is not able to achieve a more accurate result on this uni-modal function, F_1. This is because the PBO contains only two bacteria in a population. For this uni-modal function which has a flat valley, the two bacteria can only reach a limited accuracy of the minimum within a limited number of

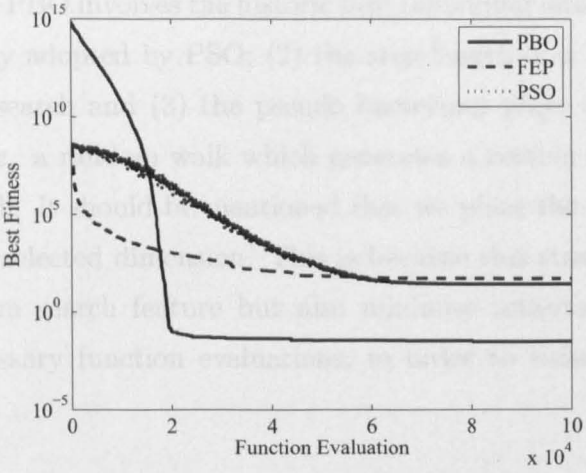


Figure 3.18: The convergence process for f_3

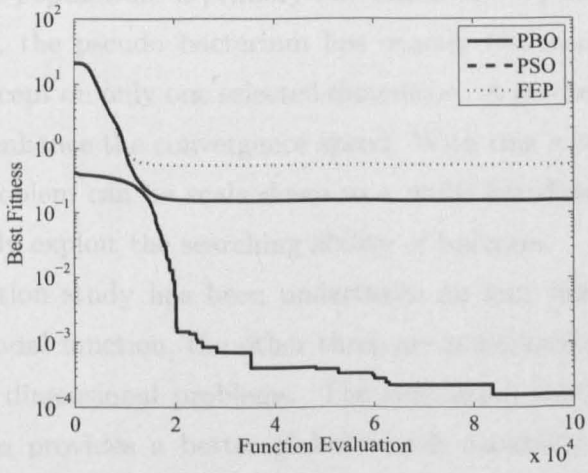


Figure 3.19: The convergence process for f_4

iterations. However, PBO performs much better than PSO and FEP on the multi-modal functions, $F_2 \sim F_4$, in terms of the number of function evaluations and search accuracy. Figures 3.16 $\sim$ 3.19 demonstrate the convergence process obtained in each function evaluation of the three algorithms during 100,000 evaluations. The merits of PBO are represented by its characteristics which

include that (1) PBO involves the historic best individual into the computation, in the same way adopted by PSO; (2) the step length of a bacterium is used to guide local search and (3) the pseudo bacterium plays an important role in mutation, *i.e.* a random walk which generates a certain range of diversity for global search. It should be mentioned that we place the pseudo bacterium on a randomly selected dimension. This is because this strategy can not only create a random search feature but also minimise uncertainties which may lead to unnecessary function evaluations, in order to reduce the amount of computation.

3.6.3 Discussion

A very simple BaIA, PBO, has been presented, which contains only two individuals in a population: a primary bacterium and a pseudo bacterium. In every iteration, the pseudo bacterium has exactly the same position as the primary one except on only one selected dimension. A gradient-based search is introduced to enhance the convergence speed. With this methodology, a high-dimensional problem can be scale down to a multi low-dimensional problem, thus, it can fully exploit the searching ability of bacteria.

The simulation study has been undertaken on four benchmark function, one is a uni-modal function, the other three are multi-modal functions, all of them are high dimensional problems. The simulation results show that this novel algorithm provides a better global search capability and convergence performance, given the same or lpwer evaluation numbers of the objective function, in comparison with PSO and FEP. The less computation load is one of the major merits of the algorithm.

3.7 Summary

In this chapter, a new branch of BIA - BaIA, has been introduced. Two preliminary works have been presented in detail. This original study demonstrates paradigm of optimisation techniques and will provide a foundation for algo-

rithm development. The three BaIAs , BFAVP, BSA and PBO have been proposed to tackle optimisation problems from three different perspective. BFAVP was developed to demonstrate the knowledge transfer from models to algorithms: how to incorporate the bacterial foraging patterns, *e.g.* chemotaxis and metabolism, into the optimisation algorithms, and enhance its searching ability. A further knowledge transfer from biology to optimisation can be found in BSA, in which the study of group behaviour of bacterial foraging, (*i.e.* quorum sensing) has been concentrated on. The factors of attraction and dispersion were introduced to make sure the balance between global exploration and local search was achieved. Finally, PBO was investigated with the aim of reducing computational load. In PBO, computation is based on individual coordinates instead of a combination of coordinates. Thus, a pseudo-gradient-based local search can be performed on a specific direction, which outperforms the other strategies in terms of convergency speed, given a certain level of randomness. All the simulation studies have been undertaken to evaluate the developed algorithms on the benchmark problems. The simulation results have demonstrated their effectiveness in solving complex optimisation problems.

Chapter 4

Bacteria-inspired Algorithms For Global Optimisation in Varying Environments

4.1 Introduction

Optimisation in dynamic environments has received great attention in recent years [106]. In this case, optimisation algorithms are required to be able to continuously track a changing optimum over time. Thus, the balance between the convergence and searching ability should be carefully considered. Over the last two decades, EAs designed to solve static optimisation problems, have been comprehensively and intensively investigated, while EAs developed specifically for dynamic environments were rarely considered. In recent years, with the emergence of another member of the EA family – BFA, the self-adaptability of individuals in group searching activities has attracted a great deal of interest. In this chapter, a BFA aimed at optimisation in dynamic environments, called DBFA, is developed. A test bed proposed previously in [107] is adopted to evaluate the performance of DBFA. The simulation studies offer a range of changes in a dynamic environment and the results show that DBFA can adapt to various environmental changes which occur with different probabilities, with

both satisfactory accuracy and stability, in comparison with BFA [69].

4.2 Conventional Dynamic Evolutionary Algorithms

Static optimisation problems have been the focus of evolutionary computation for a long time. However, there are many practical problems in various fields in the real word, which need optimisation methodologies suitable for a changing environment.

In the changing environment problems, there are only a few very coarse-grained classifications distinguishing alternating (or cyclic) problems: problems with changing morphology, drifting landscapes, and abrupt and discontinuous problems. Other classifications are based on the parameters of problem generators. Many real-world applications are dynamic, *e.g.* scheduling control problems, vehicle routing, and portfolio optimisation. However, current approaches either ignore dynamics and re-optimise regularly or use very simple control rules. There are some possible remedies, such as: restart after a change (only choice if changes are too severe), but this is too slow; or generate diversity after a change. However, randomisation destroys information, only local search can be performed afterwards, which functions similar to restart. Some other algorithms maintain diversity throughout the run, however they also disturb optimisation process.

There has already been some previous research to tackle this issue [108] [109]. One of these approaches detects a change in the environment, and then adjusts the algorithm parameters to increase the diversity or probability of mutation, which, on the other hand, destroys the information gained by previous search. Another maintains a certain diversity throughout the evolutionary process, for example, by introducing random immigrants; or taking the ages of individuals into account, which nonetheless, still disturbs the optimisation process. Recently, a new algorithm, called 'Memory - Enhanced Approach' [110] [111], claimed to cope with periodically changing environments. The

performance of this algorithm depends on memorising the history of the optimisation process and maintaining the diversity of the population. Therefore, it is only useful when the optimum reappears at an old location, and the problem of convergence still remains. Inspired by nature, there is a new group of approaches investigated recently, called 'Multi-Population Approach', which are gaining more attention [112] [113]. It proposes the use of a number of subpopulation groups for covering possible solutions, and enables itself to detect new optima by maintaining a suitable diversity. An algorithm called 'Self-organizing Scouts' is an example of this approach [108]. Furthermore, there are other ideas to deal with dynamic problems, such as 'thermodynamical GA' [114], ACO for dynamic problems [115] and varying population swarming [116]. However, the problems of conflict between convergence and diversity still exist in these algorithms.

For dynamic problems, rapid convergence, which is an important characteristic for algorithms used in static optimisation problems, is not only desired, but also the ability of finding a global optimum is required. However these two requirements are contradictory to each other. There should be a compromise between the convergence and the diversity of the algorithm designed for solving a specific problem. Some of the reported results are encouraging. However, most of these methods were evaluated in periodically changing environments or they involve intensive computation, as detection of environmental changes is required in each search step, which are either too hypothetic or unrealistic for the complexity of real-world problems.

However, the complex and organised activities exhibited in bacterial foraging patterns could inspire a new solution for dynamic problems. The underlying mechanism of the survival of bacteria, especially *E.coli* in a changing environment has been reported by researchers in the area of biological sciences [6]. Inspired by these phenomena, an optimisation algorithm called BFA was introduced in [69], which is known to be useful for applications in control [69] or parameter estimation [70]. Based on BFA, we propose a DBFA, which is especially designed to deal with dynamic optimisation problems, combining

the advantages of local search in BFA and a new selection scheme for diversity generating.

We use the moving peaks benchmark (MPB) [107] as the test bed for experiments. The performance of the DBFA is evaluated in two ways. The first is concerned with the convergence of the algorithm in random - periodical changes in an environment, which are divided into three ranges from a low probability of changes to a higher one. The second is testing a set of combinations of the algorithm parameters which are largely related to the accuracy and stability of the algorithm. All results are compared with the existing BFA [69], and show the effectiveness of DBFA for solving dynamic optimisation problems.

4.3 Dynamic Bacterial Foraging Algorithm

4.3.1 Local search

Local search of DBFA is similar to the one in BFA. Both of them are inspired by bacterial chemotaxis, which is based on the suppression of tumbles in cells that happen by chance to be moving along an upward gradient. A unit walk with random direction represents a *Tumble* and a unit walk with the same direction as the last step indicates a *Run*. After one step's move, the position of the i^{th} bacterium can be represented as:

$$\theta_i(j+1,r,l) = \theta_i(j,r,l) + C(i)\angle\phi(j) \tag{4.3.1}$$

where $\theta_i(j,r,l)$ indicates the position of the i^{th} bacterium at the j^{th} chemotactic step in the r^{th} reproductive loop of the l^{th} elimination and dispersion event; $C(i)$ is the length of a unit walk, which is set to be a constant; and $\angle\phi(j)$ is the direction angle of the j^{th} step. When its activity is *Run*, $\angle\phi(j)$ is the same as $\angle\phi(j-1)$; otherwise, $\angle\phi(j)$ is a random angle generated within a range of [0,2π].

With the activity of *Run* or *Tumble* taken at each step of the chemotaxis process, a step fitness, denoted by $J_i(j,r,l)$, will be evaluated.

4.3.2 Selection process

The performance of BFA in static environments has been reported in detail [69]. The process of chemotaxis enables bacteria to obtain a satisfactory ability to perform a local search. It is worth noticing that the individuals in BFA could converge rapidly without information sharing between themselves, which is different from most EAs.

In dynamic environments, a rapid convergence needs to be reconsidered as the environment is changing and a fast convergence may not lead to an effective trace of the global optimum in all possible directions. The reproduction process of BFA aiming to speed up the convergence is suitable in static problems, but lacks adaptation in dynamic environments. Thus, in order to compromise between rapid convergence and high diversity, we propose a dynamic bacterial foraging algorithm (DBFA) in which a selection process is introduced using a more flexible scheme to enable a better adaptability in a changing environment. The basic idea of the DBFA is to maintain a suitable diversity for global search, while the local search ability is not degraded, and changes in the environment are also considered. The scheme is described as follows:

$$J_i = \sum_{j=1}^{n} J_i(j,r) \tag{4.3.2}$$

$$rank_i = sort(J_i) \tag{4.3.3}$$

$$W_i = m\frac{(rank_i)^k}{\sum_{i=1}^{P}(rank_i)^k} + (1-m)\frac{J_i}{\sum_{i=1}^{P} J_i} \tag{4.3.4}$$

where n is the number of chemotactic steps (each step may contain a *Run* or *Tumble*) during a bacterium's life time, j is its index and P is the population size, m is the weight of diversity, and k is the exponent of $rank_i$. At r^{th} selection step, the fitness of bacterium i, J_i, is still the sum of the step fitness during its life as indicated in equation (3.5.3), which has been redefined as $J_i(j,r)$, since there are no dispersion events. Thus, those bacteria which have experienced more nutrient-rich areas are more likely to be selected as parents for the next generation. However, this fitness-domination does not help to main-

tain diversity. Therefore, the combination of the solution and rank is chosen to prevent rapid convergence and retain the adaptation ability of the DBFA for dynamic environments. This adaptation ability is improved as long as the rank of each individual functions as a fitness-independent factor. Thus, the whole population is sorted according to J_i using an operator *sort*, then $rank_i$ is allocated as the rank of bacterium i in equation (4.3.3). We introduce the parameter m, which affects the diversity, to the selection process by combining the rank of the bacterium $(rank_i)^k$ with the fitness calculation J_i. The two factors are weighted by m. The survival probability of bacterium i, W_i, is given in equation (4.3.4), and

$$\sum_{i=1}^{P} W_i = 1$$

At last, the roulette wheel selection taken from the GA literature is adopted to generate the next generation.

As diversity can be obtained in each generation, *i.e.* a chemotactic process which contains a number of chemotactic steps, the process of 'dispersion and elimination' is not considered in this algorithm. The pseudo code of DBFA is described in Table 4.1, where N_s indicates the number of selections, N_c represents the number of chemotactic steps in a bacterium's life time, *Jlast* is a temporary variable in the process of *Run* and N_r is the maximum number of steps for a single activity of *Run*.

4.4 Assessment Criteria

In our experiments, we use three different ways to evaluate the performance of DBFA. These involve determing the average best fitness found over a given period during the evolutionary process, the accuracy and the stability of the algorithm.

- Average best over a period (ABP)

  One of the most important factors in optimisation is the ability of searching for the global optimum. In a dynamic environment, comparing the

Table 4.1: Pseudo code of DBFA

Randomly initialise positions of bacteria in the domain;
FOR (Selection $r = 1 : N_s$)
  FOR (chemotactic steps per bacterium $j = 1 : N_c$)
    Calculate: Calculate the nutrient function of bacterium i as $J_i(j, r)$;
    Tumble: For bacterium i, set $J_i(j, r)$ as *Jlast*.
    Generate a random angle represented by an array Δ, where each element belongs to $[0, 1]$;
    Move to a random direction $\frac{\Delta}{\sqrt{\Delta' \times \Delta}}$ by a unit walk, the new position is calculated by equation (4.3.1);
    Start another chemotactic step.
    Run: For bacterium i, calculate the step fitness as $J_i(j, r)$. If $J_i(j, r) < Jlast$, take another unit walk of the same direction, set $J_i(j, r)$ as *Jlast*; otherwise, start another chemotactic step;
    Continue the **Run** until N_r steps before start another chemotactic step;
  END FOR (chemotactic steps)
  Sum: Set J_i as the sum of the step fitness over the life time of bacterium i using equation (4.3.2);
  Sort: Sort J_i in ascending values of fitness in the population;
  Select: Calculate the rank of bacterium i according to equation (4.3.3); Obtain W_i for bacterium i by equation (4.3.4); Select bacteria by using roulette wheel selection.
END FOR (Selection)

best solution found after a certain number of generations is not sufficient, since the optimum might be varying over time. There is an alternative for

reporting the performance of the algorithm, which averages over the best solution found at each step during a period between two environmental changes. It is concerned with an average of the best values, denoted by average best, found over a period T_i, where T_i denotes the i^{th} period. This average best over a period is denoted as ABP, which is similar to the 'Best Fitness' for static environments, but only for a given period T_i. Let S_i be the first step of T_i, E_i be the last step. Thus, ABP is defined as:

$$ABP_{T_i} = \frac{1}{E_i - S_i} \sum_{t=S_i}^{E_i} f(t)^* \tag{4.4.1}$$

where $f(t)^*$ is the best fitness found in each step, and $t = N_c \times r + j$.

- Accuracy

  To obtain the accuracy of algorithm A in function F, firstly, we calculate accuracy in each step t,

$$Accuracy_{\mathrm{F,A}}(t) = \frac{f_{\mathrm{F,A}}(t)^* - V_{\mathrm{wF,A}}(t)}{V_{\mathrm{oF,A}}(t) - V_{\mathrm{wF,A}}(t)} \tag{4.4.2}$$

  Then, the accuracy as a whole is defined as

$$Accuracy_{\mathrm{F,A}} = \frac{1}{N} \sum_{t=1}^{N} Accuracy_{\mathrm{F,A}}(t) \tag{4.4.3}$$

  where V_{w} and V_{o} are the worst and optimum value respectively, N is the number of steps.

- Stability

  Similar to the definition of the accuracy of algorithm A in function F, the stability is defined as follows:

$$Stability_{\mathrm{F,A}}(t) = Accuracy_{\mathrm{F,A}}(t) - Accuracy_{\mathrm{F,A}}(t-1) \tag{4.4.4}$$

$$Stability_{\mathrm{F,A}} = \frac{1}{N} \sum_{t=1}^{N} \max\{0, Stability_{\mathrm{F,A}}(t)\} \tag{4.4.5}$$

4.5 Simulation Studies

4.5.1 Environment setting

The experiment is set in a testing environment called Moving Peaks Benchmarks (MPB), which is also called 'DF1' [107]. The dynamic function introduced for general purposes is a 'field of cones', with an objective function defined as follows:

$$Z = -\max_{i=1,\ldots,N}\{H_i - R_i\sqrt{(X - X_i)^2 + (Y - Y_i)^2}\} \tag{4.5.1}$$

where N is the number of cones in the environment. For the i^{th} cone, H_i indicates its height, R_i is the slope control variable, and (X_i, Y_i) represent the coordinates of its centre. The initialisation of the parameters is shown in Table 4.2.

Table 4.2: Parameter settings

Parameter	Value
N	15
H_1	0
R_1	0
X_1	0.5
Y_1	0.5
H_i	[1,10]
R_i	[8,20]
X_i	[-1,1]
Y_i	[-1,1]
i	$2,\ldots,N$

The parameters listed in Table 4.2 can be adjusted to change the environment. But in our simulation studies, the height and range of slope for each cone are set to be constant, and only the positions of the cones are changing.

In this case, $x_{\text{step}}, y_{\text{step}}$ are the step sizes in the x and y directions respectively. Then, for each step i, X_{i+1} and Y_{i+1} are calculated as follows:

$$x_{\text{step}} = A_x x_{\text{step}}(1 - x_{\text{step}}) \tag{4.5.2}$$

$$y_{\text{step}} = A_y y_{\text{step}}(1 - y_{\text{step}}) \tag{4.5.3}$$

$$X_{i+1} = X_i + x_{\text{step}} Dx_i \tag{4.5.4}$$

$$Y_{i+1} = Y_i + y_{\text{step}} Dy_i \tag{4.5.5}$$

where A_x and A_y are both constants. Dx_i and Dy_i can each be assigned either 1 or -1 with equal probability. An example of the dynamic environment, generated with DF1, in four steps, is illustrated in Fig. 4.1.

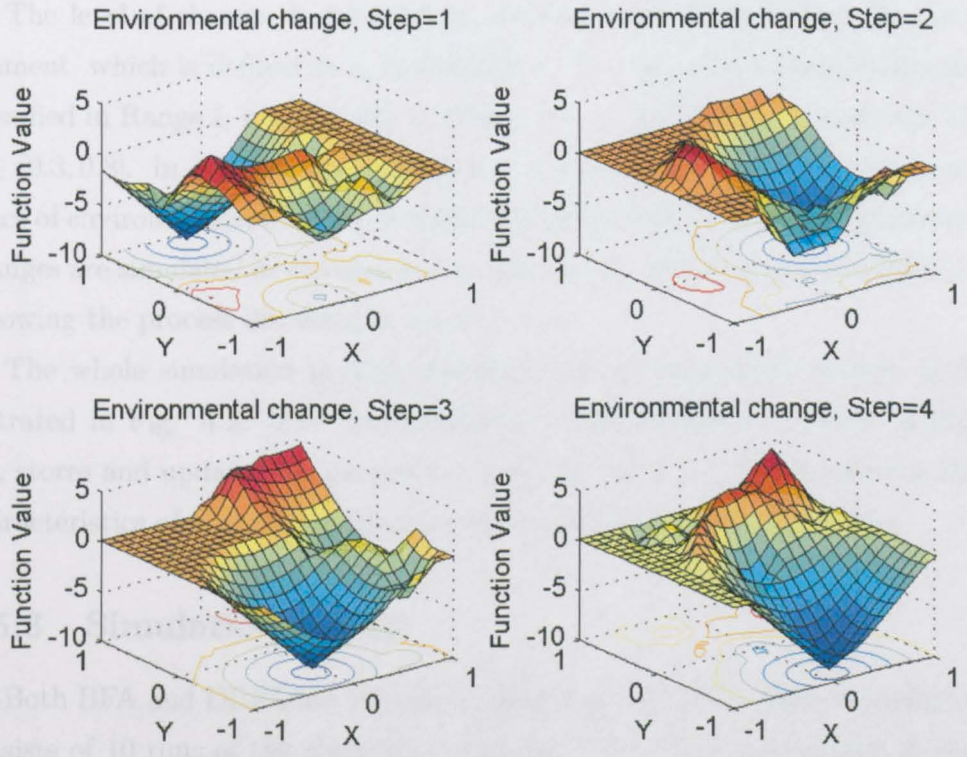


Figure 4.1: An example of environmental changes

4.5.2 Selection of DBFA parameters

The experiments are designed to 1) evaluate the adaptability of DBFA for various dynamic environments; 2) adjust the algorithm parameters (k and m in equation (4.3.4)) of the algorithm to optimise its performance. Various environmental changes are used in our simulation studies, which are divided into three ranges [117]:

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

The level of changes is reflected by the frequency of changes in the environment, which is defined as a probability τ. For the environmental changes classified in Range I, $\tau \in [0, 0.01]$, in Range II, $\tau \in [0.05, 0.2]$ and in Range III $\tau \in [0.3, 0.8]$. In our simulation studies, τ indicates the probability of occurrence of environmental changes after each chemotactic step. The environmental changes are simulated as changes in the values of X_i and Y_i in equation (4.5.1), following the process discussed in Section 4.5.1.

The whole simulation process including the environmental changes is illustrated in Fig. 4.2. The 'Environmental change database', shown in Fig. 4.2, stores and updates the parameters listed in Table 4.2, which contains the characteristics of environmental changes for evaluation of the algorithm.

4.5.3 Simulation results

Both BFA and DBFA are evaluated using the MATLAB. Each experiment consists of 10 runs of the algorithm programs. The initial parameters, k and m of the algorithm, are set as 0 and 0.5, respectively. The comparison between BFA and DBFA is given in Figs. 4.3 and 4.4, where the best fitness and ABP in a single run are plotted. To demonstrate the performance in various environments in both figures, the dynamic environment with $\tau = 0.001$, $\tau = 0.01$ and $\tau = 0.05$ are selected, which fall into to Range I and II respectively. As shown

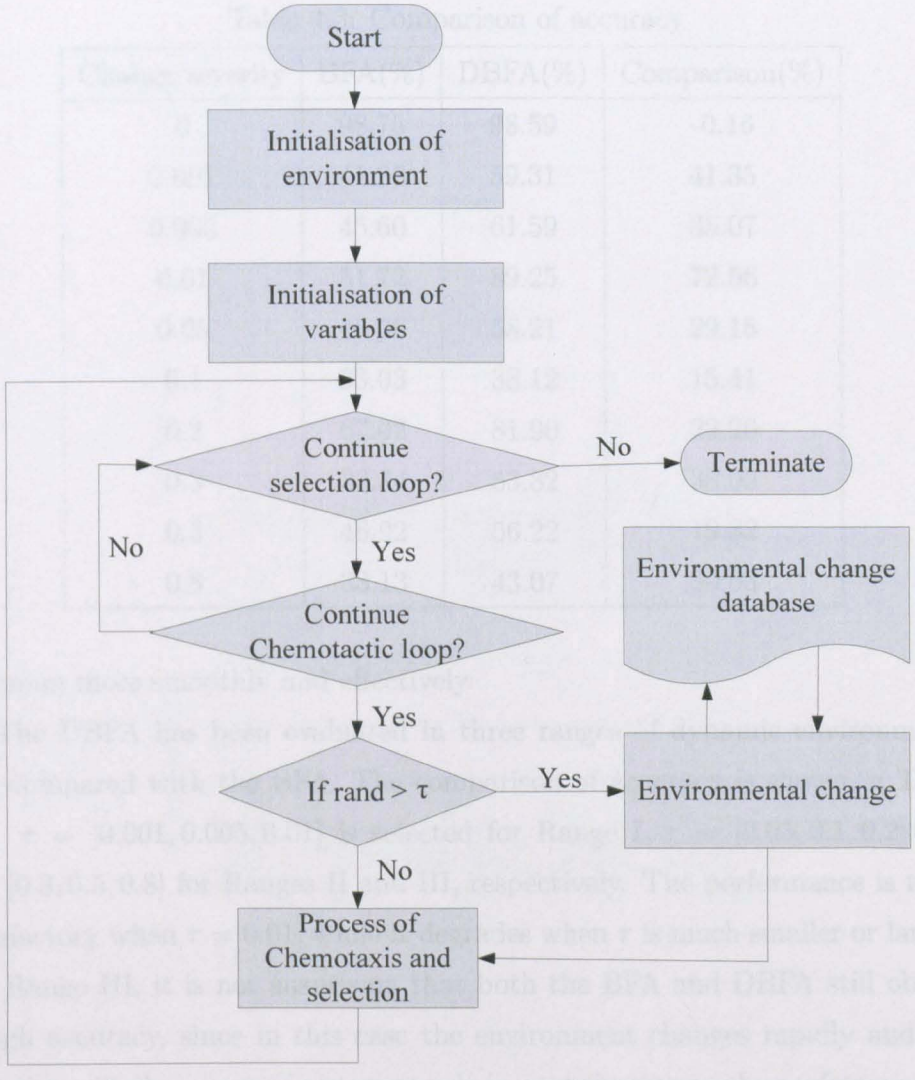


Figure 4.2: Flow chart of DBFA

in Fig. 4.3, the different parameters cause 3, 13 and 82 occurences of environmental changes in 2000 steps respectively. In this case, the performances of DBFA, for $\tau = 0.01$ and $\tau = 0.05$, are satisfactory, since the DBFA reacts to all environmental changes effectively. It is able to track 2 changes out of 3 when $\tau = 0.001$. In Fig. 4.4, the ability of searching for the optimum is evaluated by ABP. Compared with BFA, the DBFA is able to track the time-variant global

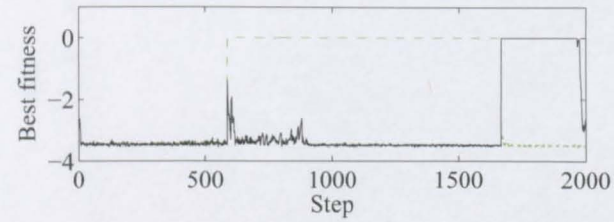
Table 4.3: Comparison of accuracy

Change severity	BFA(%)	DBFA(%)	Comparison(%)
0	98.75	98.59	-0.16
0.001	41.96	59.31	41.35
0.005	45.60	61.59	35.07
0.01	51.72	89.25	72.56
0.05	45.06	58.21	29.18
0.1	33.03	38.12	15.41
0.2	67.02	81.90	22.20
0.3	33.34	53.82	38.05
0.5	46.92	56.22	19.82
0.8	33.13	43.07	30.03

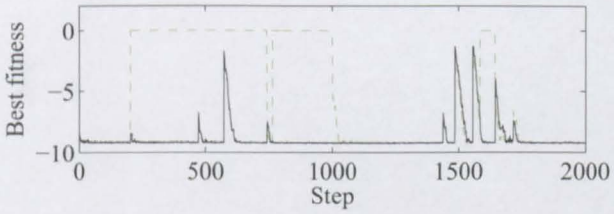
optimum more smoothly and effectively.

The DBFA has been evaluated in three ranges of dynamic environments and compared with the BFA. The comparison of accuracy is shown in Table 4.3. $\tau = [0.001, 0.005, 0.01]$ is selected for Range I, $\tau = [0.05, 0.1, 0.2]$ and $\tau = [0.3, 0.5, 0.8]$ for Ranges II and III, respectively. The performance is most satisfactory when $\tau = 0.01$, while it degrades when τ is much smaller or larger. For Range III, it is not surprising that both the BFA and DBFA still obtain a high accuracy, since in this case the environment changes rapidly and the diversity will play a more important role in contributing to the performance of the algorithm than the local search. The difference between the accuracies of BFA and DBFA in the static environment is less than 1% when $\tau = 0$, which indicates that DBFA has the almost same ability of local search.

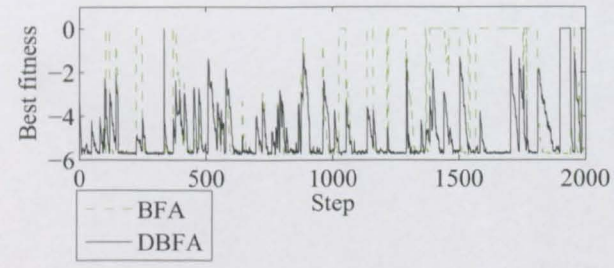
The stability in Table 4.4 is related to the accuracy, as shown in equation (4.4.4). The most desired value of stability is 0, *i.e.* the smaller, the better. It is illustrated in the table that the values of the comparisons of the two algorithms almost decreases monotonically as the severity of changing environment increases, which demonstrates that DBFA becomes stabler than BFA when the environment changes more frequently.



(a) τ=0.001



(b) τ=0.01

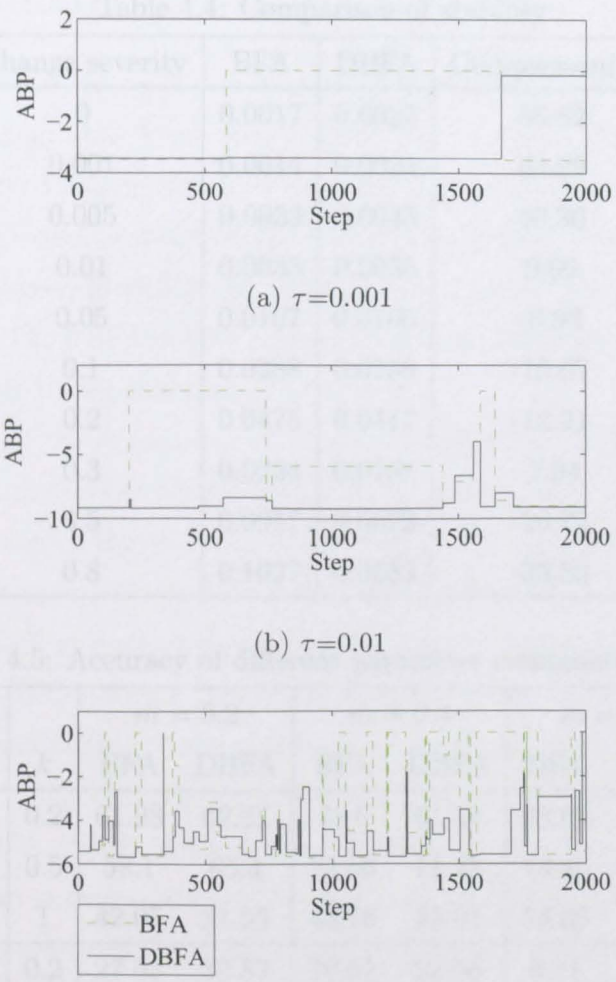


(c) τ=0.05

Figure 4.3: Best fitness in the dynamic environment

4.5.4 Discussion

To a large extent, the performance of DBFA depends on the two parameters, m and k, as given in equation (4.3.4). In order to obtain a better performance of DBFA, we evaluated the DBFA with $k = \{0.2, 0.5, 1\}$ and $m = \{0.2, 0.4, 0.7\}$ instead of $k = 0$ and $m = 0.5$ as the initial values. The DBFA was also tested with a set of environmental parameters with $\tau = \{0.005, 0.1, 0.5\}$ respectively. We compare the average accuracies of BFA and DBFA for 10 runs. The results



(a) τ=0.001

(b) τ=0.01

(c) τ=0.05

Figure 4.4: ABP in the dynamic environment

are shown in Table 4.5.

The DBFA with $m = 0.7$ performs worse than when $m = 0.2$ and $m = 0.4$, respectively. It is also worse than the BFA in 4 cases out of 9 as listed in Table 4.5, although it is slightly better in the other 2 cases. However, in general, it can be seen that for the 2 cases when $m = 0.2$, $\tau = 0.005$ and $m = 0.4$, $\tau = 0.1$, the DBFA offers the best performance. Table 4.5 indicates that when $k = 0.2$, the DBFA has the stablest performance. In this situation, the accuracy of

Table 4.4: Comparison of stability

Change severity	BFA	DBFA	Comparison(%)
0	0.0017	0.0027	58.82
0.001	0.0014	0.0023	64.29
0.005	0.0033	0.0043	30.30
0.01	0.0033	0.0036	9.09
0.05	0.0107	0.0106	-0.93
0.1	0.0288	0.0259	-10.07
0.2	0.0475	0.0417	-12.21
0.3	0.0204	0.0198	-2.94
0.5	0.0917	0.0672	-26.72
0.8	0.1027	0.0683	-33.50

Table 4.5: Accuracy of different parameter combinations(%)

		$m = 0.2$		$m = 0.4$		$m = 0.7$	
τ	k	BFA	DBFA	BFA	DBFA	BFA	DBFA
0.005	0.2	61.98	69.21	45.6	61.59	28.65	21.54
	0.5	58.1	65.4	23.56	11.48	13.46	6.43
	1	42.07	31.95	43.76	25.07	54.63	62.00
0.1	0.2	27.08	39.87	20.51	22.56	8.21	5.63
	0.5	32.32	40.77	37.21	53.25	19.53	18.41
	1	22.12	23.42	63.24	71.94	52.77	59.53
0.5	0.2	64.66	79.07	31.48	36.40	21.54	21.67
	0.5	69.13	76.51	26.74	26.32	42.60	56.44
	1	65.27	73.43	56.65	66.11	30.34	30.97

DBFA is better than that of BFA in all cases except the ones when $m = 0.7$, where the convergence has been disturbed by a high diversity.

4.6 Summary

A new DBFA for optimisation in dynamic environments, based on bacterial foraging behaviour, has been presented in this chapter. The existing BFA employs the basic foraging activities to mimic 'chemotaxis' and uses an artificial reproduction process to speed up the convergence process. Due to the lack of the balance between diversity and convergence, it is not suitable for dynamic environments. The DBFA adopts a selection scheme which enables the bacteria to flexibly adapt to the changing environment.

We have used the dynamic environment generated by DF1 to evaluate the DBFA, and compared the DBFA with BFA in three aspects: the average best over a period, algorithm accuracy and stability. The simulation results show that in all three ranges of environmental changes, the DBFA is able to provide a satisfactory performance, and can react to most of the environmental changes in time. The selection of the DBFA parameters has also been discussed.

It is worth mentioning that the diversity of DBFA changes after each chemotactic process, unlike the dispersion adopted by the BFA after several generations. The DBFA utilises not only the local search but also applies a flexible selection scheme to maintain a suitable diversity during the whole evolutionary process. It outperforms BFA in almost all dynamic environments tested in the simulation studies. The DBFA has the same computational complexity as the BFA but offers a better performance, although the issue of computation is not discussed in this chapter.

It should also be mentioned that the BFA and DBFA stemmed from a background which is totally different from that of the evolutionary computation techniques including GA and PSO, *etc.* The BaIAs are still in the process of being investigated and they are not mature yet. However, from the understanding of their essential behaviour, we can see the potential of this kind of methodology, as demonstrated in this chapter and Passino's work [69]. On the other hand, it has been understood that GA and PSO were developed specifically for static optimisation problems, although they have also been attempted for use in dynamic optimisation problems. However, we have formally

proposed, for the first time, the BaIAs for dynamic optimisation problems.

Chapter 5

Applications of Bacteria-inspired Algorithms

5.1 Introduction

OPF problem is concerned with the aim of calculating power flow and optimising power system operating conditions within specific constraints. The OPF problem has already been attempted as a static optimisation problem, by adopting conventional gradient-based methods and more recently, non-conventional ones, such as EAs. This chapter presents the work of solving OPF with BFAVP. The simulation results have demonstrated the merits of BFAVP in tackling the OPF problem.

On the other hand, as the loads, generation capacities and network connections in a power system are always changing, these static-oriented methods are of limited use for this issue. This chapter presents how to use DBFA to solve the OPF problem in a dynamic environment in which system loads are changing. DBFA has been evaluated for optimising the power system fuel cost with the OPF embedded, on the standard IEEE 30-bus and 118-bus test systems, respectively, with a range of load changes which were governed by different probabilities. The simulation results show that DBFA can more rapidly adapt to load changes, and more closely trace the global optimum of the system fuel

cost, in comparison with BFA and PSO.

Apart from the OPF problem, this chapter also discusses the application of BSA to improve modelling of transformer winding based on Frequency Response Analysis. With the the aim of deriving the parameters of a well-known lumped parameter model of transformer winding, a model-based identification approach is proposed. BSA is utilised in performing optimisation. Simulation results and discussions are presented to explore the potentials of the proposed approach.

5.2 OPF in Static Form

5.2.1 Background

The OPF problem has been well studied over the past a few decades [118][119][120]. Many optimisation techniques have been applied to solve this problem, which are mainly gradient-based techniques such as nonlinear programming. These gradient-based solutions have made considerable contributions to power system operation and control. However, these techniques do not overcome the difficulties in solving complex cost functions which are not differentiable in particular in the case of a high dimension or when more complicated constraints are applied.

In recent years, EAs, such as GA and PSO have been proposed for complex optimisation problems to avoid the problems of non-differentiable objective functions and local optima solutions. These algorithms have been applied to the OPF problem [41][121], but since they are based on stochastic search in population and generations, they are computationally intensive and time consuming. Although they have been well investigated in academic studies, few of them have been used in industrial power systems, since a huge amount of computation is involved in solving an OPF problem for a real power system which contains thousands of nodes and constraints, and a limited time period is given for online OPF computation. Therefore, recently, parallel computation of EP applied to the OPF has been proposed [122].

In this section, BFAVP was adopted to solve the OPF problem and has been evaluated on a practical OPF problem, focusing on minimising the fuel cost of a power system, in comparison with PSO. The evaluation of the algorithm was carried out using an IEEE 30-bus power system. The simulation results are reported to show the merits of this algorithm.

5.2.2 OPF formulation

The OPF problem can be treated as a constrained optimisation problem. The requirement of a steady state performance of a power system could be represented by an objective function with several equality and inequality constraints. Based on the conventional problem statement of power flow [123], OPF can be mathematically formulated as follows :

$$\min \quad F(\mathbf{x}, \mathbf{u}) \tag{5.2.1}$$

$$\text{s.t.} \quad g(\mathbf{x}, \mathbf{u}) = 0 \tag{5.2.2}$$

$$h(\mathbf{x}, \mathbf{u}) \leq 0 \tag{5.2.3}$$

$\mathbf{x}$ is the vector of dependent variables, which can be given as:

$$\mathbf{x}^T = [P_{G_1}, V_{L_1} \cdots V_{L_{N_L}}, Q_{G_1} \cdots Q_{G_{N_G}}, S_1 \cdots S_{N_E}] \tag{5.2.4}$$

where P_{G_1} is the slack bus power, V_L the load bus voltage, Q_G the generator reactive power output and S_k the apparent power in branch k.

$\mathbf{u}$ is a set of the control variables, which can be expressed as:

$$\mathbf{u}^T = [P_{G_2} \cdots P_{G_{N_G}}, V_{G_1} \cdots V_{G_{N_G}}, T_1 \cdots T_{N_T}, Q_{c_1} \cdots Q_{c_{N_C}}] \tag{5.2.5}$$

where P_G is the generator real power outputs except that at the slack bus, V_G the generator voltages, T the transformer tap setting and Q_c the reactive power generations of VAR sources.

The equality constraints $g(\mathbf{x}, \mathbf{u})$ are the nonlinear power flow equations which can be formulated as follows:

$$0 = P_{G_i} - P_{D_i} - V_i \sum_{j \in N_i} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}) \quad i \in N_0 \tag{5.2.6}$$

$$0 = Q_{G_i} - Q_{D_i} - V_i \sum_{j \in N_i} V_j (G_{ij} \sin\theta_{ij} + B_{ij} \cos\theta_{ij}) \quad i \in N_{PQ} \tag{5.2.7}$$

The inequality constraints $h(\mathbf{x}, \mathbf{u})$ are the limits of the control variables and state variables which can be expressed as:

$$\begin{aligned} P_{G_i}^{\min} \le P_{G_i} \le P_{G_i}^{\max} \quad & i \in N_G \\ Q_{G_i}^{\min} \le Q_{G_i} \le Q_{G_i}^{\max} \quad & i \in N_G \\ Q_{C_i}^{\min} \le Q_{C_i} \le Q_{C_i}^{\max} \quad & i \in N_C \\ T_k^{\min} \le T_k \le T_k^{\max} \quad & i \in N_T \\ V_i^{\min} \le V_i \le V_i^{\max} \quad & i \in N_B \\ |S_k| \le S_k^{\max} \quad & i \in N_E \end{aligned} \tag{5.2.8}$$

One approach tackling a constrained optimisation problem is to turn it into an unconstrained one, with special terms added to penalise equality and inequality constraint violations [124]. Thus, the objective function equation (5.2.9) is generalised as follows:

$$\begin{aligned} J = F + \lambda_P \left(P_{G_1} - P_{G_1}^{\lim}\right)^2 + \sum_{i \in N_V^{\lim}} \lambda_{V_i} \left(V_i - V_i^{\lim}\right)^2 + \\ \sum_{i \in N_Q^{\lim}} \lambda_{G_i} \left(Q_{G_i} - Q_{G_i}^{\lim}\right)^2 + \sum_{i \in N_E^{\lim}} \lambda_{S_i} \left(|S_i| - S_i^{\max}\right)^2 \end{aligned} \tag{5.2.9}$$

where λ_P, λ_{V_i}, λ_{G_i} and λ_{S_i} are the penalty coefficients. The limit value $x_i^{\lim} (x \in (P_G, V, Q_G, S))$ can be defined as

$$x_i^{\lim} = \begin{cases} x_i^{\max} & \text{if} \quad x_i > x_i^{\max} \\ x_i^{\min} & \text{if} \quad x_i < x_i^{\min} \end{cases} \tag{5.2.10}$$

5.2.3 Pseudo code of BFAVP-OPF algorithm

The pseudo code of BFAVP-OPF algorithm is listed in Table 5.1.

Table 5.1: Pseudo code of the BFAVP-OPF

Set $k = 0$;	
Randomly initialise positions of bacteria;	
Initialise the objective function with J, *e.g.* fuel cost ;	
WHILE (termination conditions are not met)	
FOR(each bacterium p)	
	Tumble: Generate a random tumbling angle φ_p^k. Move to the new direction by equation (3.4.1), and draw energy $\tilde{e}_p^k$ from the environment by equation (3.4.7);
	Run: For the p^{th} bacterium, calculate the evaluation value in the current step. If the current evaluation value is less than the value obtained in the last step, the bacterium will continue to move with an angle φ_p^k until it reaches the maximal step limit N_c. It also draw energy $\tilde{e}_p^k$ from the environment by equation (3.4.7);
END FOR	
Reproduction:	Select those bacteria which obtained enough energy to survive.
Elimination:	Calculate the lifespan for each bacterium using equation (3.4.11). Those individuals which exceed the limit of lifespan will be eliminated.
$k = k + 1$.	
END WHILE	

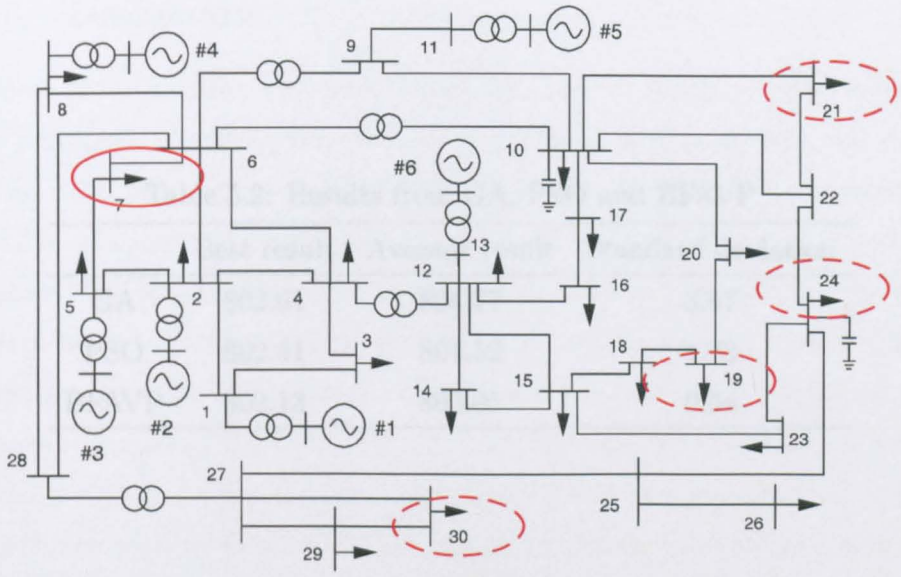


Figure 5.1: IEEE 30-bus system with the indications of the varying load buses

5.2.4 Simulation studies

The proposed BFAVP-OPF was tested on the standard IEEE 30-bus test system shown in Fig. 5.1. The system consists of 48 branches, 6 generator-buses, and 22 load-buses. The generators are at buses 1, 2, 5, 8, 11 and 13. Branches (6, 9), (6, 10), (4, 12) and (27, 28) contain transformers with off-nominal tap ratios. The transformer tap setting can take 17 discrete values in the range of $[0.9, 1.1]$ with a step size of 0.0125. Data for this model were obtained from [121]. This problem has been tackled using both a gradient-based optimisation method [125] and evolutionary algorithms [126] [127]. The best result, average results and standard deviation obtained by GA [126], PSO [54] and BFAVP [127] are listed in Table 5.2, respectively. The parameters adopted for PSO are those suggested by Clerc *et al* [55]. Note that the initial population size is set as 50 in all three algorithms, however, since the population in BFAVP is varying, the maximal population size can reach 150 in BFAVP.

Table 5.2: Results from GA, PSO and BFAVP

	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

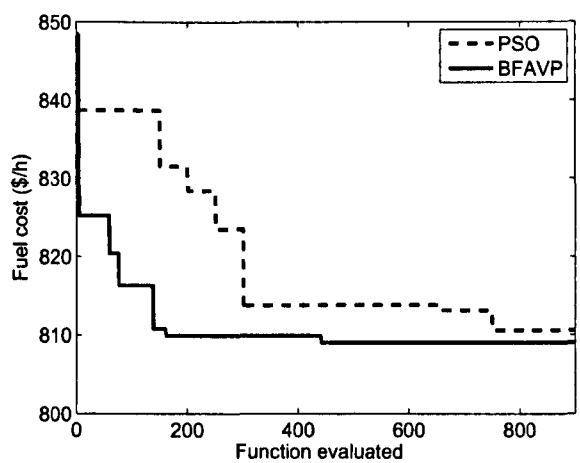


Figure 5.2: Average fuel cost for all individuals

5.2.5 Discussion

Some of the existing EAs have been attempted in recent years to solve the OPF problems. However, few successful, practical applications in real power systems have been reported, as a large mount of computation is required for these EA solutions. BFAVP is able to obtain a better result, *i.e.* 0.28 and 0.48 less than that obtained by PSO and GA, respectively. Considering the OPF problem has been attempted for over 50 years, and there have been a handful algorithms developed to tackle it, BFAVP got the best result compared with the ones reported in the literature. A more remarkable improvement can be spotted in terms of standard deviation. BFAVP has demonstrated a satisfying performance with robustness. The convergence processes of BFAVP and PSO in the first 1000 function evaluation are shown in Fig. 5.2. Compared with PSO, BFAVP is able to converge rapidly from the beginning of evaluation.

5.3 OPF With Varying Loads

5.3.1 Background

The OPF has been investigated based on many assumptions made on power system operation conditions and load patterns. However, in practice, power system operations are always subject to uncertainties, since the loads, generation capacities and network connections in a power system are always changing from time to 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. Nonetheless, in the past decades, an assumption has been made that no change is applied to power system states during the period of the online OPF computation, and the results obtained from this online computation task are accurate enough to be used for many other tasks of power system operation

such as network and generation control and system dispatch, *etc.* However, an assessment of the errors caused by this assumption has never been attempted.

Moreover, modern power systems have started to involve more distributed generations and renewable energies, such as wind power, and tide power. These relatively distributed generations will interact with transmission network frequently and make power flow in a changing state within a range from seconds to minutes. In this sense, the above assumption on power flow computation will no longer be acceptable.

In order to waive the above assumption, developing a methodology which could be used to trace the optimum solution of power flow in a dynamic environment is desired, but it is not an easy task. First of all, to consider various load changes, disturbances and power system control actions which could take place within a short period of time from seconds to minutes, the objective function of OPF cannot be assumed static and differentiable with the existence of only a single optimum. For this multi-modal optimisation problem in which the objective function is time variant, gradient-based methods cannot be used.

Currently, it has been assumed that during the computation process of online OPF, the power system is static as aforementioned; or during the variation process of power flow, the computation process of online OPF can be ignored. It can be seen that both assumptions are not reasonable. Therefore, it is desired to develop a novel optimisation algorithm which enables the optimisation calculation to proceed at the same time while the power flow changes. In other words, we need to develop an adaptive optimisation algorithm.

It is well known that EAs can be used for multi-modal optimisation problems. However, there are still a few problems if EAs are used to solve the dynamic OPF. The current EAs are population-based approaches, mainly using best-to-survive criteria. The optimisation process is convergence oriented, and it lacks the ability of effectively responding to environmental changes. In an EA, the whole population shares the same information of the global optima, which are found from the current generation, to reproduce a population in the next generation. This could be misleading for the EAs' performance in the

whole search process, if the environment is changing. For example, PSO can only search the global minimum based on the information of the best particle found in the current population. If the global minimum of the objective function will be less than the fitness of the current best particle, once the environment changes, the knowledge of the whole population will still be used for the next generation. However, if the global minimum is becoming bigger than the minimum fitness obtained, the use of the best particle will mislead the reproduction of the next generation. In this case, the memory of the global minimum in each particle will be wrongly updated, consequently PSO will corrupt.

This section presents the application of DBFA to the OPF problem in the dynamic environment of a power system. In this environment, the variations of power loads are simulated as regular and irregular environmental changes, which is undertaken based on both IEEE 30-bus and 118-bus test systems, respectively. As the concept of OPF in dynamic environments is new and has not been investigated, it is difficult to compare DBFA with most of the existing optimisation algorithms which were delicately developed for static optimisation problems. Therefore, we compare the performance of DBFA with BFA, which is the only existing bacterial foraging algorithm, and PSO, which is a most recently developed and widely used EA. The simulation results of the fuel cost minimisation show that the proposed DBFA based OPF algorithm has a better performance than that of the BFA and PSO.

5.3.2 Optimal power flow in dynamic environments

OPF in dynamic environments can be revised as follows:

$$\min \quad F(\mathbf{x}, \mathbf{u}, \mathbf{d}(t)) \tag{5.3.1}$$

$$\text{s.t.} \quad g(\mathbf{x}; \mathbf{u}, \mathbf{d}(t)) = 0 \tag{5.3.2}$$

$$h(\mathbf{x}, \mathbf{u}) \leq 0 \tag{5.3.3}$$

$\mathbf{x}$ and $\mathbf{u}$ are the same with equations (5.2.4) and (5.2.5), respectively. $\mathbf{d}(t)$

is the disturbance vector, which can be expressed as

$$\mathbf{d}^T(t) = [d_1(t), d_2(t), \cdots, d_{N_0}(t)] \tag{5.3.4}$$

The equality constraints $g(\mathbf{x}, \mathbf{u}, \mathbf{d}(t))$ are the nonlinear power flow equations with the same form of (5.2.6).

However, in the dynamic environment, P_{D_i} in equation (5.2.6) varies from time to time, which could be calculated as

$$P_{D_i} = P_{D_i}^0 + d_i(t), \quad i = 1, \ldots, N_0 \tag{5.3.5}$$

where $P_{D_i}^0$ is the demand active power at bus i without disturbance. Generally, the most common method for solving nonlinear constrained optimisation problems is transforming a constrained optimisation problem into an unconstrained one using penalty functions.

Thus, the objective function equation is in the same form with (5.2.9).

5.3.3 Pseudo code of DBFA for dynamic OPF problem

The pseudo code of DBFA for dynamic OPF problem is described in Table 5.3, where N_s indicates the number of selections, *Jlast* is a temporary variable in the process of *Run* and N_r is the maximum number of steps for a single activity of *Run*, the number of step for *Tumble* is one.

The flowchart of the whole process is shown in Fig. 5.3.

5.3.4 Simulation studies

The proposed DBFA was also tested on the standard IEEE 30-bus test system shown in Fig. 5.1, where the load variations were applied. The ellipses in Fig. 5.1 indicate the buses where the demanded active power is varying with probability.

Dynamic environment settings

The objective of this simulation is to minimise the total fuel cost, which can be represented as:

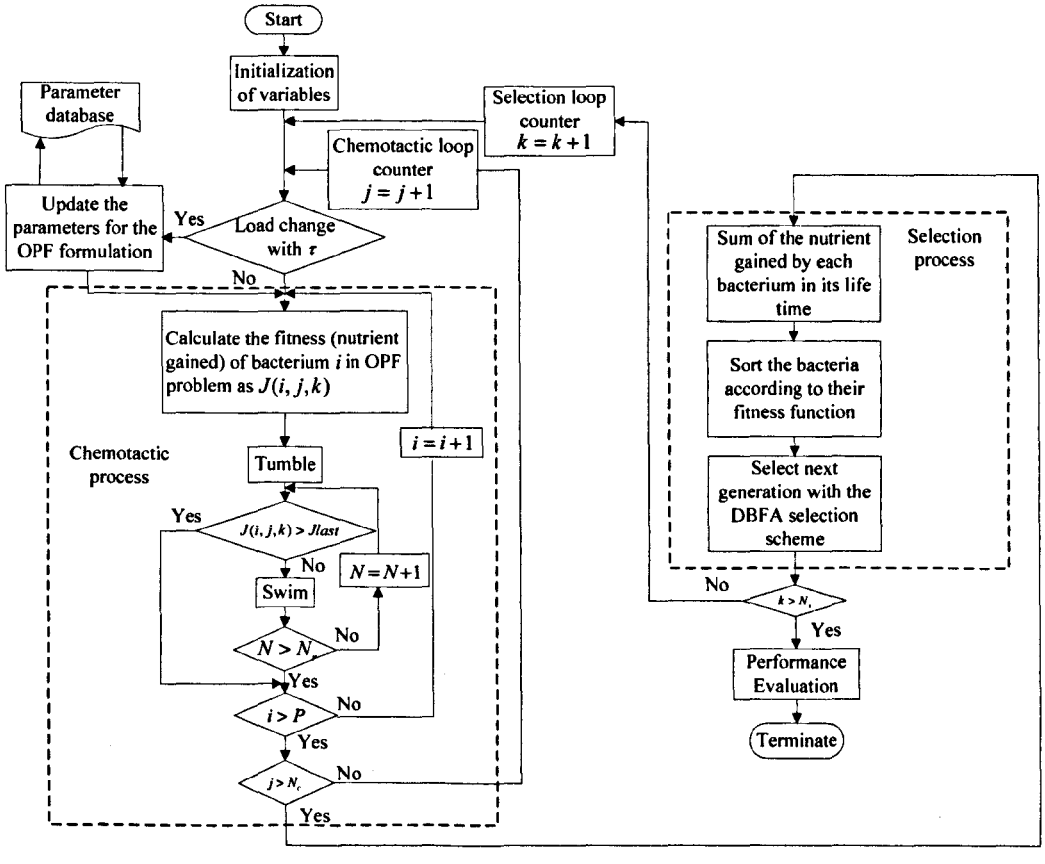


Figure 5.3: The flowchart of DBFA for dynamic OPF problem

Table 5.3: Pseudo code of DBFA for OPF problem

Randomly initialise positions of bacteria in the domain, which is represented by **u**;

FOR (Selection $k = 1 : N_s$)

FOR (chemotactic steps per bacterium $j = 1 : N_c$)

Calculate: Calculate the nutrient function **F** of bacterium i as $J(i, j, k)$;

Tumble: For bacterium i, set the initial nutrient as *Jlast*. Generate a random angle, $\phi(j)$, ranging from 0° to 360°, which indicates the new direction;
Move to the new direction by a unit walk, the new position is calculated by equation (3.5.1);

Run: For bacterium i, calculate the nutrient function $J(i, j, k)$. If $J(i, j, k) < Jlast$, take another unit walk of the same direction, set J as *Jlast*; otherwise, return to **Tumble**;
Continue the **Run** with a maximum of N_r steps;

END FOR (chemotactic steps)

Sum: Set F_i as the sum of the nutrient concentrations over the life time of bacterium i using equation (3.5.3);

Sort: Sort F_i in ascending values of fitness in the population;

Select: Calculate the rank of bacterium i according to equation (4.3.3); Obtain W_i for bacterium i by equation (4.3.4); Select bacteria using roulette wheel selection.

END FOR (Selection)

$$F = \sum_{i=1}^{N_G} f_i \tag{5.3.6}$$

where f_i is the fuel cost (\$ /h) of the i^{th} generator:

$$f_i = a_i + b_i P_{G_i} + c_i P_{G_i}^2 \tag{5.3.7}$$

a_i, b_i, and c_i are fuel cost coefficients, P_{G_i} is the real power output generated by the i^{th} generator.

In order to evaluate the performance of DBFA in dynamic OPF problems, various environmental changes are applied in our simulation studies, which are divided into three ranges [117]:

- Range I – Low level of environmental changes
- Range II – Intermediate level of environmental changes
- Range III – High level of environmental changes

The level of environmental changes can be assessed on two aspects, frequency and magnitude of changes applied on buses. The frequency can be recorded at the time instants those changes happened, however, the total effect of the changes in different or same magnitudes is difficult to estimate, since a number of changes could happen at different buses simultaneously or at different time instants, which would affect the OPF solution jointly during the period of environmental changes. In order to introduce a measure to assess the level of environmental changes and the degree of the influence of the load changes applied to the power systems (consequently leading to an assessment of the tracking ability of the algorithm, which is the major concern of this simulation study), the same magnitude of load changes was assumed to be applied on all buses.

The level of changes is reflected by the frequency of changes in the environment, which is defined as a probability τ. For the environmental changes classified in Range I, $\tau \in [0, 0.005]$, in Range II, $\tau \in [0.005, 0.01]$ and in Range III, $\tau \in [0.01, 0.05]$. In our simulation studies, τ indicates the probability of occurrence of environmental changes after each chemotactic step. The environmental changes are simulated with P_{D_7} varying by 20% of $P_{D_7}^0$. In this case,

equation (5.3.5) can be calculated as:

$$d_i(t) = \begin{cases} \alpha P_{D_i}, \alpha \in [-0.2, 0.2] & : \text{ if } i = 7 \\ 0 & : \text{ if } i \neq 7 \end{cases} \tag{5.3.8}$$

Tracking error of cost variations

In the experiment, we evaluate the performance of DBFA by concentrating on its ability to track the global optimum. We denote the tracking error of DBFA as E_{L}. Therefore, the tracking error of the cost variations in the whole process of DBFA is expressed as:

$$E_{\mathrm{L}} = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} \frac{F^*(t) - F_{\mathrm{IDEAL}}(t)}{F_{\mathrm{IDEAL}}(t)} dt \tag{5.3.9}$$

where $F^*(t)$ is the estimated cost at time t and $F_{\mathrm{IDEAL}}(t)$ is the ideal cost at time t. t_0 and t_f are the start and final time instants of the DBFA process, respectively.

Parameter settings

In DBFA, each bacterium is represented by $\mathbf{x}$ with a multi-dimensional form, as indicated in equation (5.2.4). The objective function is given by equation (5.2.9), including the fuel cost F represented by equation (5.3.6). To demonstrate the effectiveness of DBFA, BFA and PSO [54] are chosen to be compared in various cases in this study. The population in both algorithms is set to be 50. This is because this size of population has been widely used in the population-based EAs and also it is used for the purpose of comparison with the PSO which will be discussed later. The number of chemotactic steps in a bacterium's lifetime, N_{c}, and the maximum swim steps, N_{r}, for BFA and DBFA are set to be 50 and 10 by trial and error, respectively. We have noted that the results are not sensitive to these parameters. The reproduction time, N_{re}, in BFA was allowed to be unfixed, since the length of chemotaxis process depends on the variation of environment. In the selection process of DBFA, N_{s} is unfixed as well. In equation (4.3.4), the parameter n is the exponent of the ranks for each individual and m is used to allocate weights for balancing the

ranking and fitness. In this simulation study, n and m are selected to be 0 and 0.5, respectively. The standard PSO was also used in the comparison study.

The original algorithm can only handle continuous variables. However, in power systems, many control variables are discrete such as transformer tap positions. In order to handle discrete variables, the method proposed in [128] is employed in DBFA. In this case, the i^{th} member X_i contains n_{C} continuous variables and n_{D} discrete variables, and the j^{th} discrete variable is chosen from a set of m_{j} discrete values, which is expressed as:

$$X_{i,j}^{D} = [x_{i,j,1}^{d}, \cdots, x_{i,j,l}^{d}, \cdots, x_{i,j,m_j}^{d}] \tag{5.3.10}$$

When updating the variables in the optimisation process, the index of the discrete variable rather than the variable itself is involved directly in the updating process. For the j^{th} discrete variable, a fictitious real variable x is used instead of the discrete variable $x_{i,j}^{D}$, where $x \in [1, m_j + 1]$ and it is updated directly in the same way as the continuous variables in the algorithm. Then, the index l of the chosen value is determined by setting $l = INT(x)$, where $INT(x)$ denotes the greatest integer less than the real value x, to select a discrete value $x_{i,j,l}^{d}$ of the j^{th} discrete variable $X_{i,j}^{D}$, before involving it in the function evaluation.

Hence, the fitness function of the i^{th} member X_i can be expressed as follows:

$$f(X_i) \qquad i = 1, \cdots, M \tag{5.3.11}$$

where

$$\begin{aligned} X_i \;=\; & \{x_{i,j}^{c}, x_{i,j,l}^{d} \,|\, x_{i,j}^{c} \in X_i^{C}, j = 1, \cdots, n_C, \\ & x_{i,j,l}^{d} \in X_{i,j}^{D}, l \in [1, m_j],\ j = 1, \cdots, n_D\} \end{aligned} \tag{5.3.12}$$

and $X_i^{C} \in \mathbb{R}^{n_C}$, $X_i^{D} \in \mathbb{R}^{\sum_{j=1}^{n_D} m_j}$, X_i^{C} and X_i^{D} denote the feasible subsets of comprising continuous and discrete variables of member X_i, respectively.

Simulation results

BFA, DBFA and PSO are evaluated using the MATLAB. In our simulation studies, we consider firstly that the demanding active power varies at a single

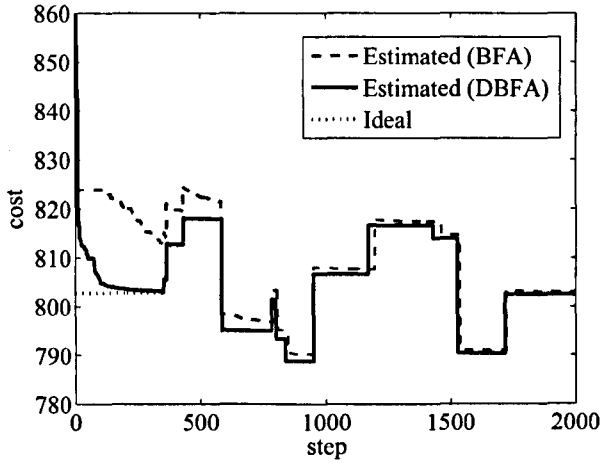


Figure 5.4: Fuel cost with the variation of load at bus 7 ($\tau = 0.005$)

bus, *i.e.* bus 7. We provide an ideal fuel cost for the comparison purpose and also assume that the global optimum can be achieved by an optimisation algorithm immediately after load changes, which is taken as the ideal cost. In our case, the ideal cost is obtained using PSO which calculates the optimal objective function as a static optimisation problem in the new environment in which the values of load changes have been updated.

The performances of BFA, DBFA and PSO with the load changes and their comparison are given in Figs. 5.4~5.6. There is no significant difference among the results obtained by PSO in the three cases, so the performance of PSO is only illustrated in Fig. 5.6. The results shown in Fig. 5.4 were obtained from the OPF in a dynamic environment with load changes at a change rate $\tau = 0.005$, which belongs to Range I and implies that there are 10 changes in a period of 2000 steps. In this case, DBFA converges much faster than BFA at the early stage, which is deemed as a static environment before the first environmental change occurs at step 370. From the first change, DBFA reacts to all the changes in time while BFA always has a delay in responding.

The significant difference in performance between BFA and DBFA can be observed in Fig. 5.5, where $\tau = 0.01$, which belongs to Range II. DBFA tracks

the changes in all cases while BFA is not able to react to most changes in time. Compared with DBFA, BFA responds these changes with a delay, and it cannot obtain a satisfactory tracing performance rapidly. The difference is represented by the big gap between the two curves.

The results obtained with $\tau = 0.05$, which belongs to Range III, are shown in Fig. 5.6. The environmental changes are considered to be more severe. In this case, it is difficult for an optimisation algorithm to respond to the changes in high frequency, since there is a limited time for an algorithm to converge to the global minimum before next change occurs. However, DBFA is able to track the global minimum and performs better than BFA. In this case, PSO fails to respond to the changes most of the time, with a constantly decreasing performance.

It has been demonstrated in Fig. 5.6 that DBFA is able to track the changes more rapidly than other algorithms, and PSO fails to converge in some cases. This is because the PSO was developed delicately for static optimisation problems. In the case of static optimisation, the global optimum found in each generation is used as a selection reference to reproduce individuals of the next generation. However, in a dynamic environment, the information of global optimum may not be necessarily useful in guiding the selection, as the real optimum of the environment is varying from time to time. Thus, as mentioned in Section 5.3.1, the information obtained from a generation is of limited use in the next generation in the dynamic environment. In this sense, due to the lack of memory updating mechanisms or appropriate diversity, PSO can only respond correctly to the environmental change in which the global minimum is less than the one in previous generation. Obviously, this is not practical.

To further demonstrate the tracking ability of DBFA, the tracking errors of both BFA and DBFA obtained from the three cases of cost variation are listed in Table 5.4. In comparison with BFA, the errors generated by DBFA are significantly reduced. In these three cases, the errors of DBFA are 19.6%, 15.9% and 38.8% of those obtained by BFA, respectively. We have also found that PSO has completely failed at all levels of load changes, due to the reason

Table 5.4: Results of tracking errors E_L

Load change mode	τ	BFA(%)	DBFA(%)	PSO
	0.005	3.275	0.642	fail
Single-bus load change	0.01	5.640	0.895	fail
Bus 7	0.05	30.587	11.864	fail
Multi-bus load changes				
Buses 7, 19, 21, 24, 30	0.2	27.132	11.071	fail

mentioned in Section 5.3.1 that PSO is only suitable for static optimisation rather than dynamic problems.

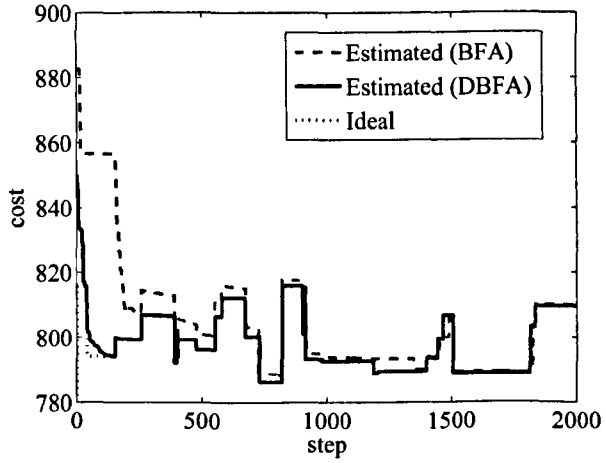


Figure 5.5: Fuel cost with the variation of load at bus 7 ($\tau = 0.01$)

Application to multi-bus load variation

In BFA, the total number of reproduction steps, N_{re}, is fixed, and a fixed N_{re} will most likely lead to the failure of computation in the dynamic OPF application. Therefore, N_{re} for DBFA and BFA is an uncertain number for solving dynamic problems. It depends on the frequencies of environmental changes. As soon as the environment changes, a reproduction occurs to prevent the algorithm from converging to the previous optimum.

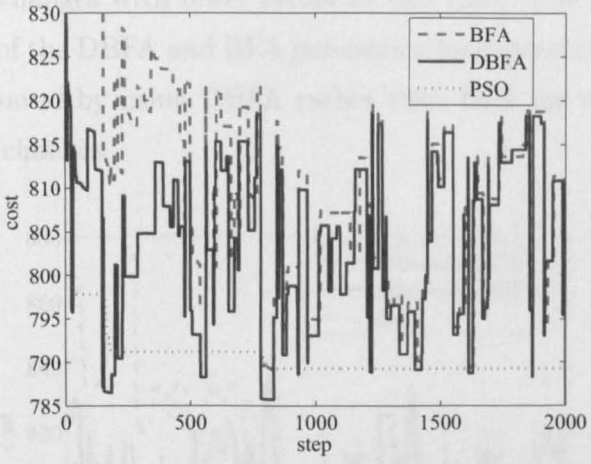


Figure 5.6: Fuel cost with the variation of load at bus 7 ($\tau = 0.05$)

- Application to IEEE 30-bus system

  The application of these algorithms is extended from a single bus load variation to multiple buses in this section. Both BFA and DBFA are tested on the same set of the standard IEEE-30 bus system with $d(t)$ applied on buses 7, 19, 21, 24 and 30, as indicated by the ellipses in Fig. 5.1. These loads vary by 20% of the rated load with $\tau = 0.2$ for each of these buses.

  The performance of both algorithms in the first 200 steps is shown in Fig. 5.7, in which the ideal cost is obtained by off-line calculation of the fuel cost for OPF with load variation considered as aforementioned. The results show that the curve obtained by DBFA can reveal the variation of fuel cost in line with the variations of the loads applied on a number of buses. Also, the comparison between the performances of BFA and DBFA indicates that DBFA is able to track the load changes more rapidly and effectively.

  The tracking errors resulting from this case are also listed in Table 5.4, in which it can be seen that the performance of BFA deteriorates since the bus loads are varying in a high frequency. However, DBFA is able to

trace the changes with fewer errors in this case. The area between the fuel costs of the DBFA and BFA indicates a large amount of energy which could be saved by using DBFA rather than BFA for the OPF problem with load changes.

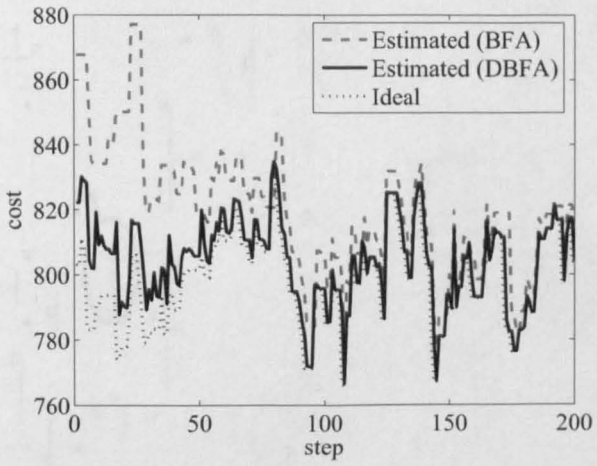


Figure 5.7: Fuel cost with the variation of multi-bus load changes ($\tau = 0.2$)

- Application to IEEE 118-bus system

  The IEEE 118-bus Test Case represents a portion of the American Electric Power System in the Midwestern US. It has 54 generator buses, 64 load buses and 186 transmission lines of which nine branches are with tap setting transformers. To demonstrate the track ability of DBFA, load variations are applied randomly with probability of $\tau = 0.2$ respectively on five buses, which are buses 11, 45, 60, 78 and 82, as indicated in Fig. 5.8 with circles.

  Figure 5.9 shows the minimal fuel cost achieved by DBFA in comparison with BFA in this dynamic environment. It should be mentioned that this figure does not include the convergence process for either DBFA or BFA. In other words, the simulated variation of the loads was applied after these two algorithms converged to a solution of OPF. In Fig. 5.9,

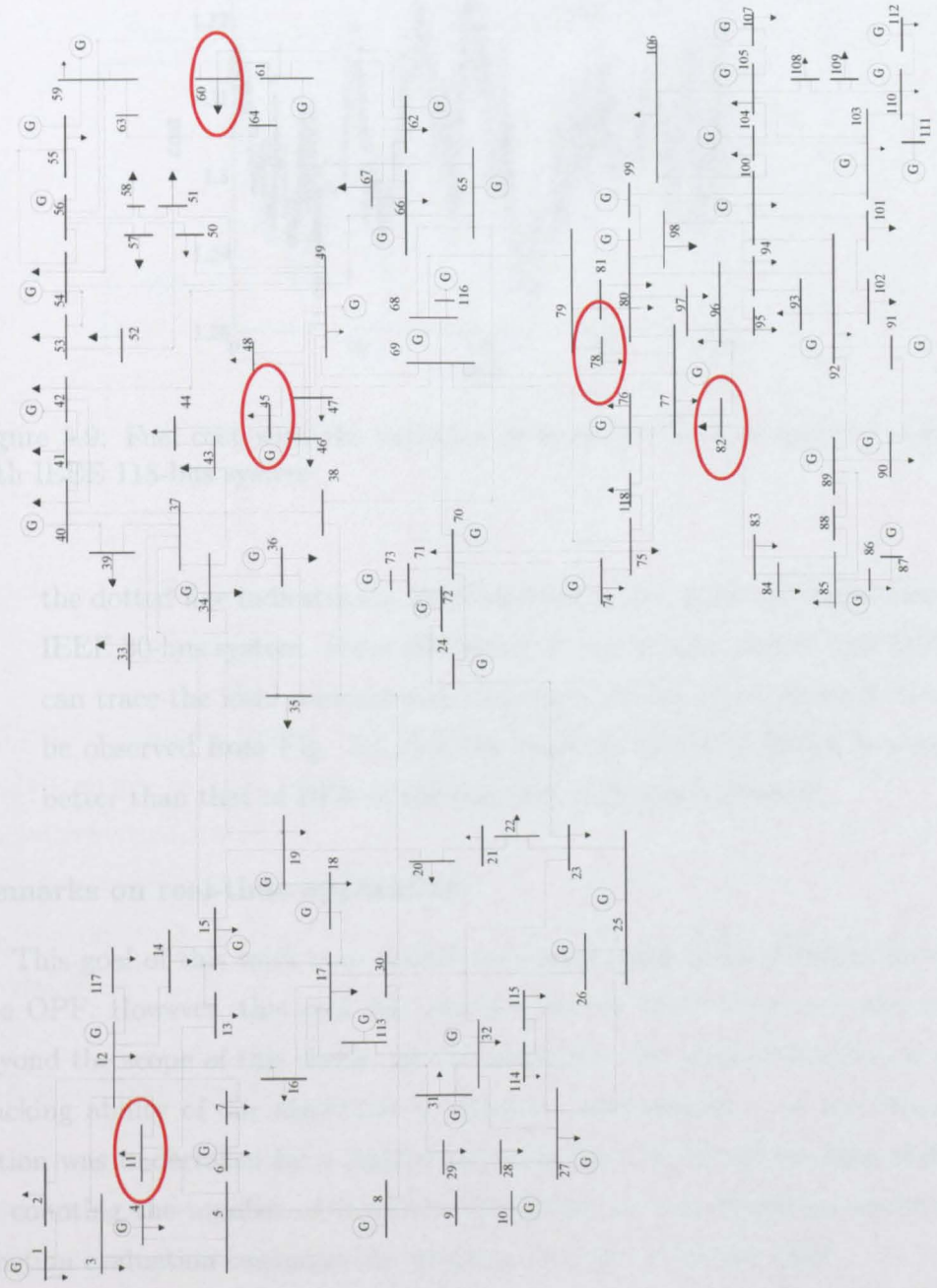


Figure 5.8: IEEE 118-bus test system with varying loads

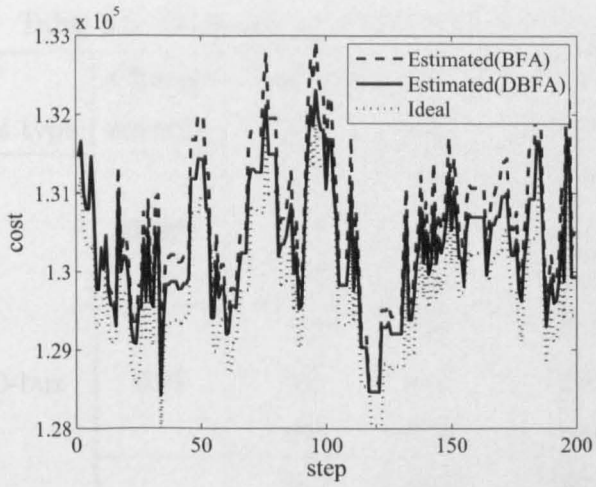


Figure 5.9: Fuel cost with the variation of multi-bus load changes ($\tau = 0.2$) with IEEE 118-bus system

the dotted line indicates the ideal solution as that achieved in the case of IEEE 30-bus system. From this figure, it can be seen clearly that DBFA can trace the ideal solution over time more closely than the BFA. It can be observed from Fig. 5.9 that the tracking ability of DBFA is always better than that of BFA as the function evaluation proceeds.

Remarks on real-time application

This goal of this work is to achieve real-time application of DBFA for on-line OPF. However, this goal still requires further effort to achieve and it is beyond the scope of this thesis. In the meantime, the research focuses on the tracking ability of our algorithm in dynamic environments. As this investigation was undertaken by a simulation study, we discuss the tracking ability by counting the number of function evaluations of each algorithm, since the function evaluation consumes the majority of time for on-line OPF.

With reference to the ideal solution, we count the total number of evaluations from a start point of load variation to the point at which 0.2%, 0.5 % and 0.8 % real-time tracking errors E_L, has been reached for each algorithm, respectively. The average numbers of function evaluations required for achieving

Table 5.5: Comparison of CPU time required

Bus type	Change severity	Real-time E_L	Function evaluation BFA	DBFA
30-bus	0.005	0.2	162	164
		0.5	151	139
		0.8	108	73
	0.01	0.2	731	508
		0.5	600	496
		0.8	331	246
	0.05	0.2	1938	1021
		0.5	1433	979
		0.8	1033	342
118-bus	0.2	0.2	68	66
		0.5	59	5
		0.8	8	3

the various tracking errors by BFA and DBFA are listed in Table 5.5.

It is shown in Table 5.5 that DBFA needs much lower numbers of function evaluations than the BFA to achieve the required error levels. This implies that it has a greater capability to be used in real-time applications, concerning the real-time on-line OPF achieved by DBFA, which is currently under investigation.

5.3.5 Discussion

The conventional OPF computation is based on an assumption that no changes of power system states would happen during a period of a few minutes within which online computation of OPF could complete. This assumption will no longer be reasonable in practice, since more distributed generations will be involved in modern power systems. This section has presented the application of DBFA to dynamic OPF problems. Inspired from the group searching activities of bacteria, the application of BFA to optimisation problems is flourishing

in recent years. Due to the lack of diversity, the existing BFA is not suitable for dynamic environments. The DBFA adopts a new selection scheme which enables the bacteria to flexibly adapt to dynamic environments, which provides a more practical solution to the OPF problems.

We have simulated the dynamic environment of the OPF problems with three different levels of load changes with probability and compared the DBFA with both BFA and PSO in this section. We have also introduced an error tracking criterion to evaluate the performance of DBFA. The simulation results demonstrate that in all three ranges of environmental changes, the DBFA is able to provide satisfactory performance, and can trace most of the environmental changes rapidly. The results also show that a significant amount of energy can be saved if the DBFA is used for solving an OPF problem with load changes, in comparison with the existing EAs and BFA.

5.4 Modelling of Power Transformer Winding

5.4.1 Background

Monitoring in-service behaviour of power transformers has become an important issue in power systems. Among various techniques applied to power transformer condition monitoring, frequency response analysis (FRA) has proved to be suitable for reliable winding displacement as well as deformation assessment and monitoring [129]. Diagnosis of these conditions through FRA relies on correct interpretation of the complex FRA results. However, direct and manual comparison between two FRA test results is the current approach used by maintenance engineers. Measured FRA traces are compared with the references taken from the same winding during previous tests or from the corresponding winding of a 'sister' transformer, or from other phases of the same transformer. The shifts in resonant frequencies and magnitude of FRA traces are believed to be indicators of a potential winding deformation. Nonetheless, the question of potential deformation location in a winding still needs to be investigated [130].

In the past a few years, there were a range of research activities being conducted, considering the simplified equivalent model of transformer winding of observing the model behaviour in frequency domain [131][132]. However, they only allow the estimation of the total approximate values of the parameters and do not consider the evaluation of winding deformation. The fundamental problem of the inaccurate simulation of transformer winding frequency behaviour is the estimation of internal parameters. Modelling of a real winding in order to obtain frequency responses, being close to experimental ones, is an extremely complex task since a detailed transformer model must consider each turn or section of a winding separately. The reason is the fluctuation of real winding parameters such as inductances and resistances per section length as well as intersection capacitances. The insulation property deviation with the frequency should also be taken into account.

A wide range of models have emerged recently to tackle this problem, such as the works done in [130][133][134][135], they require pre-knowledge of the transformer, *e.g.* demanding additional experimental tests and knowledge of geometric and physical characteristics of the transformer. However, in industry measurements it is not always possible to conduct additional tests for precise measurements of transformer geometry or insulation parameter estimation.

Recently, intelligent evolutionary learning techniques were utilised to overcome such difficulties, that offers a way to identify model parameters using available measurement data. For instance, GAs and PSO were employed for transformer thermal model parameter identification [136][137], simplified transformer winding models parameter identification [138][139], *etc.* In this section, we consider a model-based identification approach using BSA, which demonstrates a fast convergence speed and a higher accuracy, in comparison with GA.

5.4.2 Lumped parameter winding model

The basic measurement circuit is shown in Fig. 5.10. The tested impedance, *i.e.* the transformer winding, is Z_T, on which the FRA is based. The standard-

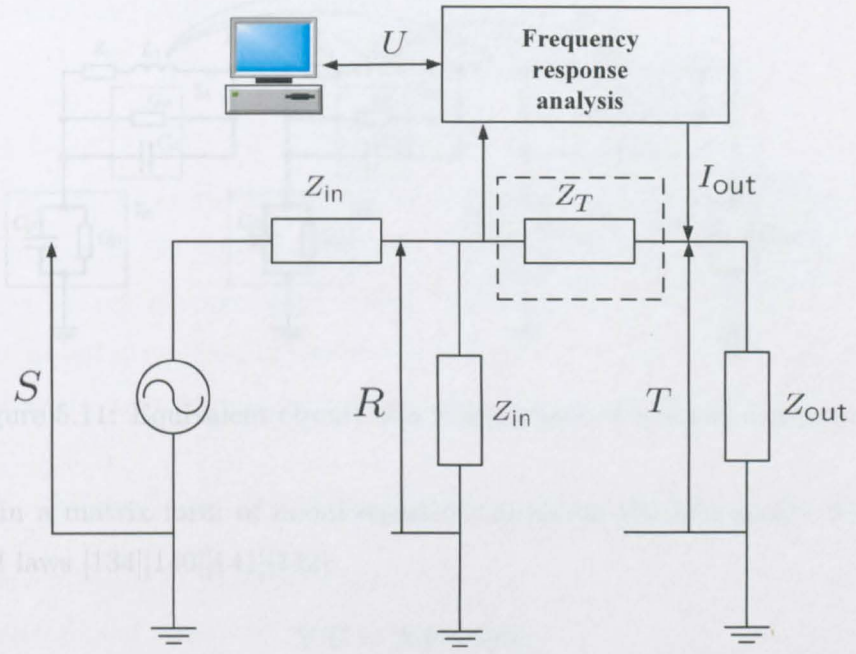


Figure 5.10: Basic measurement circuit

ised test impedance, *i.e.* the impedances of the input and output measurement cables are Z_{in} and Z_{out}, respectively. The injected signal is S, the reference measurement is R and the test measurement is T.

There are two ways of injecting the wide range of frequencies necessary, either by injecting an impulse into the winding or by making a frequency sweep using a sinusoidal signal, which is also known as swept frequency method [129]. We adopt the latter one in our work.

As the resonances of an FRA trace are related to the values of capacitances and inductances within a transformer winding, lumped-parameter equivalent circuit models have been widely utilised to analyse frequency domain behaviour [134][140][141][142][135]. The equivalent circuit of Z_T is illustrated in Fig. 5.11 . Each section of an equivalent circuit usually represents one or a few discs in the case of a disc type windings as well as one or a few turns for helical type windings [140][141][142].

The mathematical description of the model in frequency domain is usually

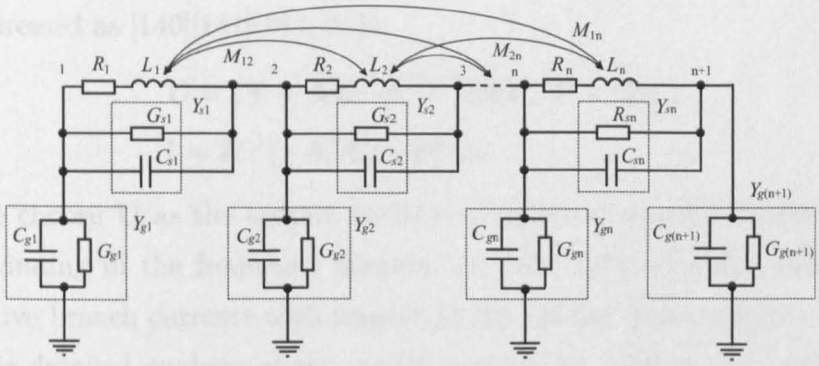


Figure 5.11: Equivalent circuit of a single phase of a power transformer

given in a matrix form of nodal equations applying the Kirchhoff's first and second laws [134][140][141][142]:

$$\begin{aligned} \mathbf{Y}\,\mathbf{U} &= \mathbf{A}\,\mathbf{I} + \mathbf{Q}U_0, \\ \mathbf{Z}\,\mathbf{I} &= -\mathbf{A}^T\mathbf{U} + \mathbf{P}U_0, \end{aligned} \tag{5.4.1}$$

where vectors $\mathbf{U}$ and $\mathbf{I}$ represent node voltages and branch currents respectively and voltage U_0 denotes an input sinusoidal signal .

Each element of the admittance matrix $\mathbf{Y}$ is a combination of admittances Y_s and Y_g, corresponding to $G_s - C_s$ and $G_g - C_g$ parallel branches respectively as shown in Fig. 5.11. Using matrix notations for the capacitance and conductance matrices, $\mathbf{Y} = j\omega\,\mathbf{C} + \mathbf{G}$, where ω is an angular frequency and j denotes the imaginary operator. The branch impedance matrix $\mathbf{Z} = j\omega\,\mathbf{L} + \mathbf{R}$ consists of the self- and mutual sectional inductances L and M, being combined into the matrix $\mathbf{L}$, and equivalent section resistances R, building the matrix $\mathbf{R}$ [134].

The incidence matrix $\mathbf{A}$ consists of -1, 0 and 1 and serves to link nodal voltages with branch currents. Matrices $\mathbf{Q}$ and $\mathbf{P}$ are formed as a component vector of $\mathbf{Y}$ and $\mathbf{A}$ matrices correspondingly in accordance with terminal connections of the winding model such as external voltage source imposition, node grounding, internodal connections, *etc.* [140][141][142][143].

From equation (5.4.1), the branch currents and nodal voltages vectors can

be expressed as [140][141][142][143]:

$$\begin{aligned}\mathbf{U} &= (\mathbf{Y} + \mathbf{A}\mathbf{Z}^{-1}\mathbf{A}^T)^{-1}(\mathbf{A}\mathbf{Z}^{-1}\mathbf{P} + \mathbf{Q})U_0, \\ \mathbf{I} &= \mathbf{Z}^{-1}(-\mathbf{A}^T\mathbf{U} + \mathbf{P}U_0).\end{aligned} \tag{5.4.2}$$

We choose $\mathbf{U}$ as the output signal to obtain the transfer function $H(j\omega)$ of a winding in the frequency domain, *i.e.* the ratio of nodal voltages and inductive branch currents with respect to the applied input voltage.

The detailed analysis of the model parameters, such as the capacitances, conductances, inductances and resistances can be found in [144].

5.4.3 Model-based identification approach

The model-based learning approach is based on searching for the optimal investigated model parameters by minimising the difference, *i.e.* fitness, between reference frequency responses and simulated model outputs. It is achieved by measuring the errors between the original responses and the model outputs. Therefore, for each individual (bacterium) of a population in BSA, its total fitness value is given as follows:

$$\min \sum_{j=1}^{S} \|H_0(\omega_i) - H(\omega_i)\|, \tag{5.4.3}$$

where $H_0(\omega_i)$ and $H(\omega_i) \in \mathbb{R}^1$ are the reference and optimised frequency responses at frequency ω_i, $i = 1, \cdots, S$, where S is the number of frequency points involved in BSA optimisation.

Due to the iterative nature of evolutionary algorithms, processing a large number of data points can greatly slow down an optimisation process. In the case of FRA, frequency responses are characterised mainly by resonant and antiresonance frequencies and corresponding magnitude values. Therefore, as proposed in [139], the dimension of processed FRA data can be reduced by selection of points of resonance and antiresonance and its vicinities for more speedy analysis.

The following steps are performed for parameter identification of the transformer windings model:

- Measurement FRA data or predefined model parameters are used to obtain the reference frequency responses.
- The selected points of the generated magnitude- and phase-frequency responses are recorded as a reference dataset, being employed as training targets for BSA optimisation.
- Assuming approximate geometrical and material parameters of the tested winding are known [143][139], estimated parameter values of an utilised winding model are calculated in order to establish the possible search space for each parameter of the model.
- BSA learning is performed, in each step of which the predefined reference dataset is compared with the corresponding values of the simulated frequency responses, being produced with the parameters obtained during optimisation process, at the same frequency points.

5.4.4 Simulation result and comparison

Parameter identification with BSA

For the numerical study, the results of experimental investigation and modelling of a transformer winding presented in [143] are chosen due to its high degree of simulation accuracy in comparison with experimental measurements. The test object is a disc-type winding consisting of 60 discs with 9 turns in each disc. The ratio of an output neutral current flowing to ground and an input signal injected into the terminal end of the winding has been used to produce frequency response measurements.

Using listed model parameters [143], the frequency responses of the test transformer winding have been obtained using the above described lumped-parameter winding model, and 71 selected points are defined as a reference dataset in the present study. The FRA simulation is performed assuming a grounded winding through a measurement impedance Z_{out}. Since the model

Table 5.6: BSA Parameters

Parameter	Notation	Value
No. of bacteria in the population	p	50
No. of chemotactic steps per bacteria lifetime	N_c	5
Swim length limits when bacteria is on a gradient	N_s	6
No. of of iteration steps	N_r	5

does not allow us to directly obtain the total neutral current, it is more convenient to express it as a product of the grounded node voltage U_{n+1} and Z_{out}.

Estimation of winding parameters are performed on a double disc basis as a unit section of the lumped circuit model, since the same partition is used in [143] for parameter calculations, which are taken as a reference in Table 5.7.

Since the mutual inductances between distant sections are negligibly small in comparison with self-inductance, it is reasonable to consider solely the self-inductances and the closest 15 mutual inductances, similar to the approach proposed in [139]. In addition, DC resistance R_{dc} is of interest to calculate the frequency variable section resistance R, as well as the series and ground capacitances and loss tangents C_s, $\tan\delta_s$ and C_g, respectively, as indicated in [144]. Therefore, 20 parameters of the model in total as shown in Table 5.7 are to be investigated with BSA. Accepted search space variations for parameter identification using BSA learning are limited to be within $\pm 10\%$ for the inductive and $\pm 50\%$ ranges for the rest of the parameters from the corresponding estimated values.

The BSA optimisation parameters are selected on the basis of the previous study on bacterial foraging optimisation [103][145] and performing numerous trials with BSA parameter variation. The selected parameters are listed in Table 5.6.

Figures 5.12 and 5.13 illustrate the comparisons of the analytically estimated and BSA identified magnitude responses with respect to the reference response. Whereas in Figs. 5.14 and 5.15 the corresponding phase responses are given.

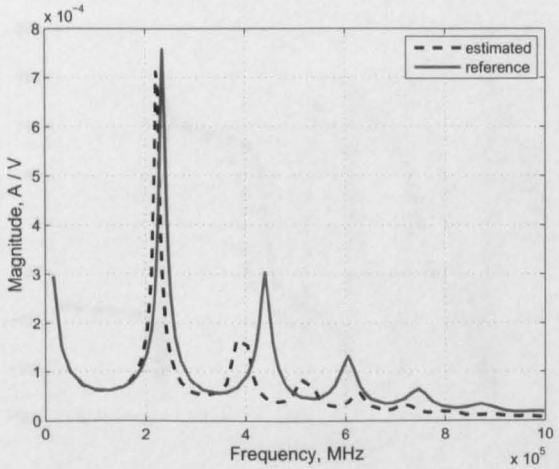


Figure 5.12: Comparison of the analytically estimated transfer function magnitude response with the reference.

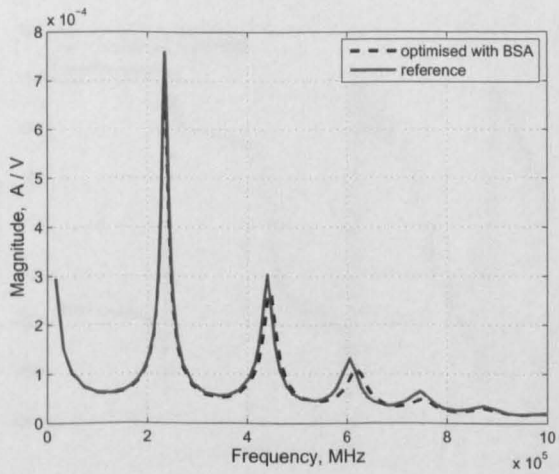


Figure 5.13: Comparison of the BSA identified transfer function magnitude response with the reference.

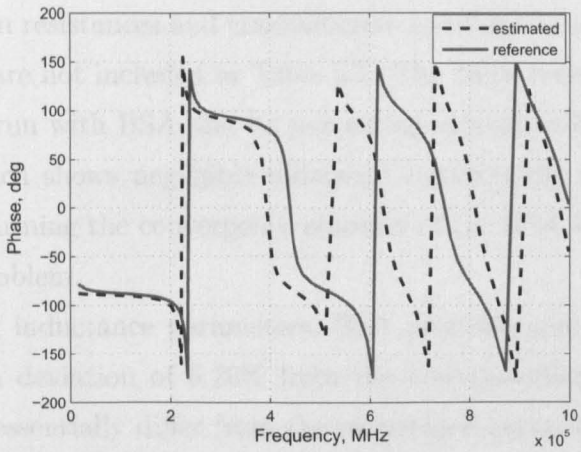


Figure 5.14: Comparison of the analytically estimated transfer function phase response with the reference.

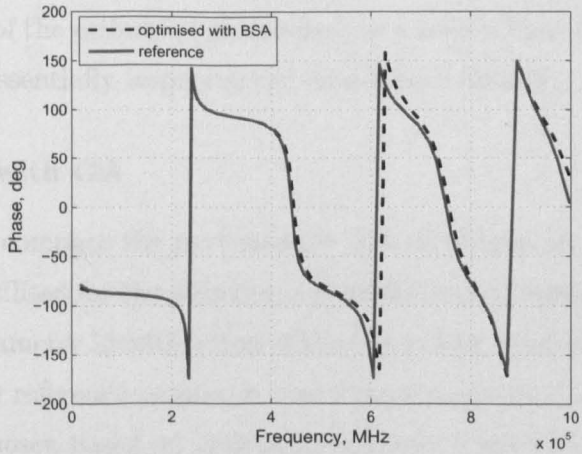


Figure 5.15: Comparison of the BSA identified transfer function phase response with the reference.

Table 5.7 summarises the reference parameter and the identified parameter values with BSA. It should be noted that the frequency dependent reference values of section resistances and conductances are given in a form of data vectors [143] and are not included in Table 5.7. The table contains the results of one successful run with BSA and its percentage deviation from the reference, analysis of which shows negligible difference between the identified parameters, thus, confirming the convergence stability of the BSA with respect to the investigated problem.

Considering inductance parameters, BSA provides accurate identification with maximum deviation of 6.26% from the corresponding reference values. This does not essentially differ from the inductance estimations which have a maximum deviation of 5.1% from the corresponding reference values. However, the major improvement of BSA identification is an adjustment of series capacitance C_s to 6.72% deviation from the reference, while the initial estimation was not successful and showed 48.69% difference. The large deviation of the initial estimation of C_s from the reference caused failure to occur at all resonance frequencies when using the model with estimated parameters.

On the other hand, in spite of a slight deviation from the reference values, the utilisation of the estimated parameters as a search basis for BSA parameter identification essentially improves the model performance.

Comparison with GA

In order to compare the performance with BSA and an evolutionary algorithm widely utilised for the parameter identification purposes, GA is employed to conduct parameter identification of the equivalent lumped parameter model using the same reference responses and fitness function (5.4.3). The GA parameters are chosen based on various preliminary trials and are listed in Table 5.8.

Table 5.7 also presents the GA identified parameters and their deviation from the reference. As seen from the table, the deviation of GA identified parameters becomes greater with respect to those obtained with BSA. For

Table 5.7: Comparison of the reference and optimised parameters of the transformer winding model

Parameter	Reference value (reprinted from [143])	Estimated	Identified		Deviation from the reference, %		
			BSA	GA	Estimated	BSA	GA
L_1, μH	186.0534	194.3	190.1	177.5	4.43	2.17	4.59
M_{12}, μH	141.4607	145.7	147.8	164.0	2.99	4.48	15.93
M_{13}, μH	91.0820	91.8	93.9	95.4	0.79	3.09	4.74
M_{14}, μH	65.3182	63.8	65.5	79.3	2.32	0.28	21.4
M_{15}, μH	48.9240	48.1	48.8	55.8	1.68	0.25	14.05
M_{16}, μH	37.5984	38.1	38.4	44.2	1.33	2.13	17.55
M_{17}, μH	29.4244	29.9	30.9	32.7	1.61	5.02	11.13
M_{18}, μH	23.3602	22.9	22.8	18.9	1.97	2.39	19.09
M_{19}, μH	18.7726	18.2	17.9	20.7	3.05	4.64	10.26
M_{110}, μH	15.2490	15.3	15.4	16.4	0.33	0.99	7.55
M_{111}, μH	12.5083	12.9	13.2	12.4	3.13	5.53	0.87
M_{112}, μH	10.3530	10.3	10.5	8.2	0.51	1.42	20.79
M_{113}, μH	8.6410	8.2	8.1	7.0	5.10	6.26	18.99
M_{114}, μH	7.2688	7.2	7.2	6.1	0.94	0.94	16.08
M_{115}, μH	6.1593	6.4	6.5	9.1	3.91	2.28	41.55
M_{116}, μH	5.2552	5.4	5.3	4.3	2.75	0.85	18.17
C_g, pF	5.0224	5.085	4.8017	4.2384	1.25	4.39	15.61
C_s, pF	85.686	127.41	79.92	87.55	48.69	6.72	2.18
R_{dc}, Ω	–	0.0151	0.0544	0.0204	–	–	–
$\tan\delta_s$	–	0.03	0.0153	0.0670	–	–	–

Table 5.8: GA Parameters

Parameter	Value
Population size	80
Selection Algorithm	*tournament*
Crossover Algorithm	*scattered*
Crossover Fraction	0.8
Mutation Algorithm	*adapt feasible*
No. of Elite Individuals	2
Search space variation from estimated values	$\pm(10 \sim 50)\%$

instance, the deviation of the GA identified mutual inductance M_{115} reaches 41.55% in comparison with only 2.28% deviation by the BSA identification. Moreover, GA gives a worse estimate of ground capacitance C_g with 15.61% deviation compared to the one identified with BSA (4.39% deviation). On the other hand, GA performs better in identification of series capacitance C_s with only 2.18% deviation against 6.72% deviation with BSA.

5.4.5 Discussion

A well-known lumped parameter model of transformer winding is utilised for FRA simulation in this section. A model-based identification approach is formulated to determine the parameters of the model with BSA. Initial search space for identification of the model parameters are established based on parameter estimated values, which are calculated using analytical expressions. The BSA has delivered a satisfactory performance during optimisation compared with the simulated reference FRA responses. The comparative study with GA reveals that BSA is more efficient for the given optimisation problem and shows better identification performance. There is a slight difference between the identified and preset parameters, which is negligible in a practical sense. The model parameter identification using experimental input admittance responses shows that the proposed approach can be further utilised for

the experimental FRA results interpretation aimed at winding fault diagnosis.

5.5 Summary

The OPF problem and modelling of power transformer winding have been regarded as complicated optimisation problems in power system. This chapter demonstrated the results of solving those problems using BaIAs, which were introduced in Chapters 3 and 4. BFAVP has been used for tackling OPF problem in static form and DBFA for the OPF in dynamic environments. These results have shown their effectiveness in tackling real-world optimisation problems in both static and dynamic forms. The modelling of power transformer winding has been attempted by BSA for parameter optimisation. The simulation results have shown its effectiveness and it can achieve the results closer to the referred parameter values than those obtained by GA.

Chapter 6

Conclusions

This chapter concludes the thesis and summarises the major achievements of the research work in bacteria-inspired modelling and optimisation. The applications of BaIAs in real-world optimisation problems of OPF and modelling of power transformer winding are also introduced. Challenges of the work are described, followed by the suggestions for future research.

BaIAs are an emerging branch of BIA, which introduces a wide range of biological features into the algorithm design. For instance, bacterial chemotaxis and QS are involved in the formulation of the algorithms. This thesis presents four newly developed BaIAs, and demonstrates their effectiveness in the benchmark testing. They have also been applied to the static and dynamic OPF problems. In particular, the new concept of optimisation tracking in a varying environment has been proposed and it has been at the first time applied to the area of power systems. The BaIA application to identification of model parameters of power transformer winding is also novel. These two applications have strongly shown the potential of the BaIAs in dealing with large-scale complex optimisation problems.

6.1 Outcomes

This study focuses on analysing and modelling bacterial foraging patterns and developing BaIAs based on the modelling work. A novel model, VE-

BAM, has been developed to simulate these patterns, such as chemotaxis and QS, *etc.* Following this model, three BaIAs, BFAVP, BSA and PBO, have been developed. Variant from the approaches developed for static optimisation problems, DBFA has been proposed specifically for dynamic optimisation problems, which has also been applied successfully to the optimal power flow problem with varying loads. The BSA has been applied to the identification of model parameters of power transformer windings. The achievements obtained from this research are summarised as follows.

A brief introduction to the recently emerged research branch, Systems Biology, was given in Chapter 1. Its relationship with BIAs has also been discussed, followed by a non-exhaustive introduction of the existing BIAs. The motivation for choosing modelling of bacteria as the departure point to study Systems Biology was given, and the most significant features exhibited in bacterial foraging behaviour were illustrated in this chapter. The outline and major contributions of this thesis were also presented.

Chapter 2 was devoted to the development of VEBAM, which is a novel model simulating bacterial foraging behaviour, especially including Chemotaxis and QS at both the individual and the population levels. This model was based on IbM, which is a tool to represent global consequences of local interactions between individuals of a population. Therefore, not only the characteristics of group patterns, such as QS, but also the individual behaviour of bacteria, *i.e.* chemotaxis and metabolism are studied and recorded. This work is significantly different from the conventional modelling in which characteristics of the population are represented statistically and individual features could only be analysed from the modelling results.

In the past decades, BIAs have flourished, with EAs as a typical representor. However, EAs share some similarities, *i.e.* a simple abstract structure that uses a few biological ideas to implement. In order to investigate a more biologically-plausible algorithm, the study of bacterial foraging patterns has attracted great attention. The survival skill of bacteria (*i.e.* searching for the food source), and in particular the patterns exhibited by *E.coli* in a time-varying environ-

ment were reported. Chapter 3 introduced two preliminary works in this area, BC and BFA, followed by three newly developed algorithms, BFAVP, BSA and PBO. BFAVP simulates not only chemotaxis but also proliferation and elimination, with simplified quorum sensing incorporated to improve convergence. BSA explores the potential of bacterial foraging from another perspective, *i.e.* chemotactic ability and group behaviour. PBO, developed with the aim of enhancing computational efficiency, markedly reduces the number of function evaluations, while generating satisfying optimisation results. The evaluation of the algorithms have been undertaken on the existing benchmark problems, with encouraging results obtained.

Apart from the static optimisation problems, a new BaIA, DBFA, developed specifically for dynamic optimisation problems was presented in Chapter 4. In dynamic environments, a rapid convergence may not lead to an effective tracing of the global optimum in all possible directions. Thus, in order to compromise between rapid convergence and high diversity, we propose DBFA in which the balance has been achieved by adopting a selection scheme assisted by a weighting system.

Chapter 5 demonstrates three applications of BaIAs in power systems, namely solving OPF problems in both static form and dynamic environments with BFAVP and DBFA, respectively; and modelling of power transformer winding with BSA. In the first application, a brief introduction of OPF problems is given, followed by the formulation of OPF problems. BFAVP was applied to solve the fuel cost minimisation problem static form and DBFA was employed to minimise the fuel cost on both the IEEE 30-bus test system and the IEEE 118-bus test system respectively, under a wide range of variations of bus loads. The results obtained by BFAVP and DBFA outperform other widely used algorithms. In the third application, the model parameters for power transformer winding were optimised by BSA, which is able to obtain a superior result than those obtained by estimated results by theoretical analysis and GA, respectively.

In summary, this thesis encompasses both the modelling of bacterial forag-

ing behaviour and the optimisation algorithms inspired by these phenomena. As a new branch of the BIA family, BaIAs benefit from the multi-disciplinary study in biological system and demonstrate great potential in solving optimisation problems. This thesis also presents the applications of these algorithms to power system problems with satisfying performance.

6.2 Challenges

Throughout the PhD study, a significant number of challenges have been overcome. In conceiving the VEBAM, the study was always in a dilemma that the whole project could face two directions, one is modelling-oriented and the other is optimisation-oriented. The two directions seem to be unrelated and they tend to follow different lines. Modelling aims at representing a biological system quantitatively, exploring the details of biological phenomena; However, BIA is a reverse process by generalising and simplifying the biological system to develop optimisation algorithms. Nevertheless, the approaches should be biologically-plausible, although most of the parameters are generated by trail and error study. One of the major challenges is to find a way to make VEBAM a reasonable analog to the real bacterial foraging system while making it possible to be transformed into an optimisation algorithm.

Furthermore, there is a long-standing problem in BIAs. As we know, the 'No Free Lunch' theorem generally applies to BIAs, therefore the performance of the developed algorithms are difficult to compare. Although there are sets of benchmarks for comparing algorithms, results are not conclusive, since a given algorithm may perform well in certain scenarios and badly in others. In order to develop more effective BaIAs, it is crucial to use methodology which is stemmed from bacterial foraging patterns. However, the existing algorithms rarely achieve this goal, which denotes the second major challenge of this project.

The third major challenge is the application of BaIAs to power systems, including the OPF and modelling of power transformer winding. With a large

set of parameters to be optimised in these problems, it is not an easy task to represent the nature of those problem accurately, such as the formulation of the OPF with varying loads, which has not been attempted before.

There have been other problems, such as recording and analysing data in the modelling work, since it is an open-ended simulation and a slight change of one parameter will cause the data simulated previously to become invalid. Another unforseen problem was the visualisation, the computation process can be dramatically slowed down if all the trajectories of a large population are printed onto the screen.

6.3 Future Work

With the aim of both modelling and optimisation work, there are many avenues down which the current work can travel.

The first avenue concerns VEBAM. The simulation of the bacterial foraging patterns contains two levels currently, *i.e.* individual and population levels. However, the study of the colony behaviour in the population level is based on one bacterial species, *i.e. E.coli.* In the future work, the simulation of the evolution of the colony could possibly take more species into account. The 'cell-cell communication' across several species could not only be a necessary part to extend VEBAM to a more complex ecological model, but also make it possible to be applied to a wider research area.

As to the detailed components, energy (*i.e.* the nutrient provided by the environment) is the only element currently included. However, this consideration is not enough for modelling of a more realistic biological system. There are many other aspects affecting the motion of bacteria, such as the distribution of oxygen and carbon. Beyond this, an unstable environment should be further considered. On the other hand, the energy consumption of the system is extremely complex. For example, the major part of energy in the environment consumed by the bacteria will be used as the 'overhead' of the movement, such as the growth of the biomass, and therefore only a small part

of the energy can be used to move. However, in the current work, the factor to transfer energy from the environment to bacteria is artificially set. Therefore, this 'metabolism' should also be considered carefully.

BFAVP, BSA and PBO may also be extended. For instance, sensory adaptation should be studied, since it enhances the adaptability of the algorithms to a wider range of optimisation problems. On the aspect of dynamic optimisation, DBFA can be refined by implementing a more effective strategy to maintain the diversity of population, which is also the key factor in tackling dynamic optimisation problems. Furthermore, there has always been an interest in taking the optimisation algorithms toward a more specific application, such as vehicle routing problems in telecommunication and financial problems.

Finally, as has been said many times, the hurdles in terms of spatial and temporal scales could be deepened to understand the real biological system. This study will no doubt enhance the application of BaIAs in engineering problems. The gap between the modelling of biological systems and the development of BaIAs for optimisation will require a large amount of work on the study of the Systems Biology.

Appendix A

Global optimisation benchmark functions

Table A.1: The 23 benchmark functions, where n is the dimension of the function, S is the feasible search space, and $f_{\min}$ is the global minimum value of the function.

	Test function	n	S	$f_{\min}$
$f_1(x)$	Sphere Model	30	$[-100, 100]^n$	0
$f_2(x)$	Schwefel's Problem 2.22	30	$[-10, 10]^n$	0
$f_3(x)$	Schwefel's Problem 1.2	30	$[-100, 100]^n$	0
$f_4(x)$	Schwefel's Problem 2.21	30	$[-100, 100]^n$	0
$f_5(x)$	Generalized Rosenbrock's Function	30	$[-30, 30]^n$	0
$f_6(x)$	Step Function	30	$[-100, 100]^n$	0
$f_7(x)$	Quartic Function with Noise	30	$[-1.28, 1.28]^n$	0
$f_8(x)$	Generalized Schwefel's Problem 2.26	30	$[-500, 500]^n$	-12569.5
$f_9(x)$	Generalized Rastrigin's Function	30	$[-5.12, 5.12]^n$	0
$f_{10}(x)$	Ackley's Function	30	$[-32, 32]^n$	0
$f_{11}(x)$	Generalized Griewank Function	30	$[-600, 600]^n$	0
$f_{12}(x)$	Generalized Penalized Function 1	30	$[-50, 50]^n$	0
$f_{13}(x)$	Generalized Penalized Function 2	30	$[-50, 50]^n$	0
$f_{14}(x)$	Shekel's Foxholes Function	2	$[-65.536, 65.536]^n$	1
$f_{15}(x)$	Kowalik's Function	4	$[-5, 5]^n$	0.0003075
$f_{16}(x)$	Six-hump Camel-Back Function	2	$[-5, 5]^n$	-1.0316285
$f_{17}(x)$	Branin Function	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18}(x)$	Goldstein-Price Function	2	$[-2, 2]^n$	3
$f_{19}(x)$	Hartman's Function 1	3	$[0, 1]^n$	-3.86
$f_{20}(x)$	Hartman's Function 2	6	$[0, 1]^n$	-3.32
$f_{21}(x)$	Shekel's Family 1	4	$[0, 10]^n$	-10
$f_{22}(x)$	Shekel's Family 2	4	$[0, 10]^n$	-10
$f_{23}(x)$	Shekel's Family 3	4	$[0, 10]^n$	-10

Sphere Model:

$$f_1(x) = \sum_{i=1}^{30} x_i^2$$

Schwefel's Problem 2.22:

$$f_2(x) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|$$

Schwefel's Problem 1.2:

$$f_3(x) = \sum_{i=1}^{30} \left(\sum_{j=1}^{i} x_j \right)^2$$

Schwefel's Problem 2.21:

$$f_4(x) = \max_i \{|x_i|, 1 \leq i \leq 30\}$$

Generalized Rosenbrock's Function:

$$f_5(x) = \sum_{i=1}^{29} (100(x_{i+1} - x_i^2)^2 + (x_i - 1))^2$$

Step Function:

$$f_6(x) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2$$

Quartic Function with Noise:

$$f_7(x) = \sum_{i=1}^{30} i x_i^4 + \text{random}[0, 1)$$

Generalized Schwefel's Problem 2.26:

$$f_8(x) = -\sum_{i=1}^{30} \left(x_i \sin\left(\sqrt{|x_i|}\right)\right)$$

Generalized Rastrigin's Function:

$$f_9(x) = \sum_{i=1}^{30} (x_i^2 - 10\cos(2\pi x_i) + 10)^2$$

Ackley's Function:

$$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos 2\pi x_i\right) + 20 + e$$

Generalized Griewank Function:

$$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{30}(x_i - 100)^2 - \prod_{i=1}^{30}\cos(\frac{x_i - 100}{\sqrt{i}}) + 1$$

Generalized Penalized Functions:

$$\begin{aligned} f_{12} &= \frac{\pi}{30}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{29}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} \\ &+ \sum_{i=1}^{30} u(x_i, 10, 100, 4) \end{aligned}$$

and

$$\begin{aligned} f_{13} &= 0.1\left\{\sin^2(\pi 3x_1) + \sum_{i=1}^{29}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})]\right. \\ &+ \left.(x_n - 1)^2[1 + \sin^2(2\pi x_{30})]\right\}\sum_{i=1}^{30} u(x_i, 5, 100, 4) \end{aligned}$$

where

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i + 1)$$

Shekel's Foxholes Function:

$$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right]^{-1}$$

where $(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}$

Kowalik's Function:

$$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$$

Six-hump Camel-Back Function:

$$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

Branin Function:

$$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$$

$$-5 \le x_1 \le 10,\ 0 \le x_2 \le 15$$

$$x_{\min} = (-3.142, 12.275),\ (3.142, 2.275),\ (9.425, 2.425)$$

$$\min(f_{17}) = 0.398$$

Table A.2: Kowalik's Function f_{15}

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

Goldstein-Price Function:

$$f_{18} = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$$
$$\times[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

$$-2 \leq x_i \leq 2 \quad \min(f_{18}) = f_{18}(0, -1) = 3$$

Hartman's Function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right]$$

with n=3,6 for $f_{19}(x)$ and $f_{20}(x)$, respectively. The coefficients are defined by Tables and , respectively.

$$0 \leq x_j \leq 1$$

$$\min(f_{19}) = f_{19}(0.114, 0.556, 0.852) = -3.86$$

$$\min(f_{20}) = f_{20}(0.201, 0.150, 0.477, 0.275, 0.311, 0.657) = -3.32$$

Table A.3: Hartman's Function f_{19}

i	a_{ij}, $j = 1,2,3$			c_i	p_{ij}, $j = 1,2,3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.038150	0.5743	0.8828

Table A.4: Hartman's Function f_{20}

i	$a_{ij},\ j=1,2,3,4,5,6$						c_i	$p_{ij},\ j=1,2,3,4,5,6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Shekel's Family:

$$f(x) = -\sum_{i=1}^{m}[(x-a_i)(x-a_i)^T + c_i]^{-1}$$

with $m = 5, 7$ and 10 for $f_{21}(x)$, $f_{22}(x)$ and $f_{23}(x)$, respectively. $0 \leq x_j \leq 10$. $x_{\text{local-opt}} \simeq a_i$ and $\min(f_{x_{\text{local-opt}}}) \simeq 1/c_i$ for $1 \leq i \leq m$.

Table A.5: Shekel's Family f_{21},f_{22},f_{23}

i	a_{ij}, $j = 1,2,3,4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.1
7	5	5	3	3	0.1
8	8	1	8	1	0.1
9	6	2	6	2	0.1
10	7	3.6	7	3.6	0.1

Appendix B

A non-exhaustive list of BIAs

Algorithm	Background and features	Authors (years)
GA	A simple model using two basic genetic operators: mutation and crossover, using fixed-length binary strings	Holland (1975)
VEGA	A multi-objective GA	Schaffer (1985)
NPGA	A multi-objective GA based on Niched Pareto-optimal solution	Horn *et al.* (1994)
EP (Cauchy mutation)	Cauchy mutation is used instead of Gaussian mutation	Yao *et al.* (1999)
(1+1) - ES	Real-valued vector is used and mutation is applied with an identical standard deviation to each object variable	Rechenberg (1973) and Schwefel (1981)
$(\mu + 1)$ - ES	A multi membered ES with $\mu > 1$ parents	Bäck (1996)
GP	Based on a tree structure to discover the computer program or functional structure	Koza (1992)
PSO	A trajectory tracking algorithm inspired by the social behaviour of bird flocking or fish schooling	Kennedy (1995)

CPSO	A constriction factor is used to ensures the convergence of the dynamic system	Clerc (2002)
PSOPC	A passive congregation behaviour is incorporated into PSO to improve global search performance	He *et al.* (2004)
CLPSO	Only historical optimum is used in the searching process Fixed-length binary strings	Liang *et al.* (2006)
ACO	Based on the study of collective foraging behaviour of ants and used for route optimisation *etc.*	Dorigo *et al.* (1997)
GSO	Based on 'Producer-Scrounger' (PS) model of general animal behaviour Fixed-length binary strings	He *et al.* (2006)
BFA	Inspired by bacterial foraging patterns	Passino (2002)
BSA	An improved BFA with introduction of quorum sensing	Tang *et al.* (2006)
BFAVP	A varying population algorithm incorporating a comprehensive model of bacterial foraging	Li *et al.* (2007)

References

[1] A. S. Fraswer. Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Science*, 10:484–499, 1957.

[2] R. M. Friedberg. A learning machine: Part i. *IBM Journal*, pages 2–13, 1958.

[3] C. M. Henry. Systems biology: Integrative approach in which scientists study pathways and networks will touch all areas of biology, including drug discoverty. *CENEAR*, 81(20):45–55, May 2003.

[4] Multi disciplinary Centre for Integrative Biology. Intergrative biology. Available Online At: http://www.mycib.ac.uk/ (accessed 24 June 2007).

[5] D. Noble. Modeling the heart. *Physiology*, 19:191–197, 2004.

[6] H. C. Berg and D. A. Brown. Chemotaxis in *escherichia coli* analyzed by three-dimensional tracking. *Nature*, 239:500–504, 1972.

[7] D. M. Fleming, D. L. DeAngelis, and L. J. *et al.* Gross. Atlss: Acrosstrophic-level system simulation for the freshwater wetlands of the everglades and big cypress swamp. Technical report, National Biological Service Technical Report, 1994.

[8] S. F. Railsback, R. H. Lamberson, B. C. Harvey, and W. E. Duffy. Movement rules for individual-based models of stream fish. *Ecological Modelling*, 123:73–89, 1999.

[9] B. Orland. Smartforest - an interactive forest data modeling and visualization tool. In *USDA Forest Service Remote Sensing Applications Conference*, Salt Lake City, 1994.

[10] G. Booth. Gecko: a continuous 2-d world for ecological modeling. *Artificial Life*, 3(3):147–163, 1997.

[11] J. Kreft, G. Booth, and J. Wimpenny. Bacsim, a simulator for individual-based modelling of bacterial colony growth. *Microbiology*, 144:3275–3287, 1998.

[12] C. W. Reynolds. Flocks, herds, and schools: a distributed vehavioral model. In *Computer Graphics (SIGGRAPH)*, volume 21, pages 25–34, 1987.

[13] Systems biology for energy and environment. Report on the computational biology workshop for the genomes to life program, U.S. Department of Energy, Germantown, Maryland, August 2001.

[14] Wolkenhauer. O., H. Kitano, and K. Cho. Systems biology: Looking at opportunities and challenges in applying systems theory to molecular and cell biology. *IEEE Control Systems Magazine*, pages 38–48, August 2003.

[15] M. Hucka, A. Finney, H. Sauro, and H Bolouri. Systems biology markup language (sbml) level 1: Structures and facilities for basic model definitions. Version 2 (final), Carlifornia Institute of Technology, Pasadena, CA, USA, August 2003.

[16] H. Kitano. Systems biology: A brief overview. *Science*, 295:562–564, 2002.

[17] R. J. Cano and J. S. Colome. *Microbiology*. West Publishing Company, St. Paul, New York, Los Angeles, San Francisco, 1986.

[18] R. Dillon, L. Fauci, and D. Gaver. A microscale model of bacterial swimming, chemotaxis and substrate transport. *Journal of Theoretical Biology*, 177:325–340, 1995.

[19] M. D. Baker, P. M Wolanin, and J. B. Stock. Signal transduction in bacterial chemotaxis. *BioEssays*, 28(1):9–22, 2005.

[20] D. Bray. Bacterial chemotaxis and the question of gain. *Proceedings of the National Academy of Sciences*, 99:7–9, 1995.

[21] S. P. Strong, B. Freedman, W. Bialek, and R. Koberle. Adaptation and optimal chemotactic strategy for *e. coli*. *Physical Review E*, 57(4):4604–4617, 1998.

[22] B. P. Ingalls, T. Yi, and P. A. Iglesias. Using control theory to study biology. *Journal of Physical Chemistry B*, 108:1143–1152, 2004.

[23] M. B. Miller and B. L. Bassler. Quorum sensing in bacteria. *Annual Review of Microbiology*, 55:165–199, 2001.

[24] H. Bremermann. The evolution of intelligence. the nervous system as a model of its environment. Contract No. 477 (17) 1, Department of Mathematics, University of Washington., 1958.

[25] J. Holland. *Adaptation in Natural 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 Publishing Company, Inc., 1989.

[27] S. Venkatraman and G. G. Yen. A generic framework for constrained optimization using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 9(4):424–435, 2005.

[28] V. K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combing the effects of variable population size and reinitialization to

enhance performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28, 2006.

[29] Y. J. Cao and Q. H. Wu. Optimisation of control parameters in genetic algorithms: A stochastic approach. *International Journal of Systems Science*, 30:551–559, 1999.

[30] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, Mahwah, NJ, USA, 1985. Lawrence Erlbaum Associates, Inc.

[31] J. Horn, N. Nafpliotis, and D. E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.

[32] K. Rodriguez-Vazquez, C. M. Fonseca, and P. J. Fleming. Identifying the structure of nonlinear dynamic systems using multiobjective genetic programming. *IEEE Transactions on Systems Man and Cybernetics Part A-Systems And Humans*, 34:531–545, 2004.

[33] Q. H. Wu, Y. J. Cao, and J. Y. Wen. Optimal reactive power dispatch using an adaptive genetic algorithm. *International Journal of Electrical Power and Energy Systems*, 20(8):563–569, 1998.

[34] T. Yamada and R. Nakano. Genetic algorithms for job-shop scheduling problems. In *UNICOM seminar, Proceedings of Modern Heuristic for Decision Support*, pages 67–81, 1997.

[35] C. Harpham, C. W. Dawson, and M. R. Brown. A review of genetic algorithms applied to training radial basis function networks. *Neural Computation and Application*, 13:193–201, 2004.

[36] M Koppen, K. Franke, and R. Vicente-Garcia. Tiny gas for image processing applications. *IEEE Computational Intelligence Magazine*, May 2006.

[37] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer - Verlag, Berlin, second, extended edition edition, 1994.

[38] L. Fogel, A. Owens, and M. Walsh. *Artificial Intelligence Through Simulated Evolution.* Wiley, New York, 1966.

[39] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, July 1999.

[40] D. Nam, Y. D. Seo, L-J. Park, C. H. Park, and B. Kim. Parameter optimization of an on-chip voltage reference circuit using evolutionary programming. *IEEE Transactions on Evolutionary Computation*, 5(4):414–421, 2001.

[41] Q. H. Wu and J. T. Ma. Power system optimal reactive dispatch using evolutionary programming. *IEEE Transactions on Power Systems*, 10(3):1243–1249, 1995.

[42] I. Rechenberg. *Evolutionstrategie: Optimieriung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, Germany, 1973.

[43] H. P. Schwefel. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[44] T. Back. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, 1996.

[45] G. Rudolph. Finite markov chain results in evolutionary computation: A tour d'horizon. *Fundam. Inform.*, 35:67–89, 1998.

[46] T. Hatanaka, K. Uosaki, H. Tanaka, and Y. Yamada. System parameter estimation by evolutionary strategy. In *SICE '96. Proceedings of the 35th SICE Annual Conference*, pages 1045–1048, 1996.

[47] A. I. Gonzlez, M. Gra, J. Ruiz Cabello, A. D'Anjou, and F. X. Albizuri. Experimental resuls of an evolution-based adaptation strategy for vq image filtering. *Inf. Sci. Inf. Comput. Sci.*, 133(3-4):249–266, 2001.

[48] T. Bergener, C. Bruckhoff, and C. Igel. Parameter optimization for visual obstacle detection using a derandomized evolution strategy. 2001.

[49] A. K. Gupta and G. W. Greenwood. Applications of evolutionary strategies to fine-grained task scheduling. *Parallel Processing Letters*, 6(4):551–561, 1996.

[50] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIET Press, 1992.

[51] H. G. Beyer and H. P. Schwefel. Evolution strategies. *Natural Computing*, 1:3–52, 2002.

[52] J. R. Koza, F. H. III Bennett, D. A. Keane, and F. Dunlap. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation*, 1(2):109–128, 1997.

[53] H. Guo and A. K. Nandi. Breast cancer diagnosis using genetic programming generated feature. *Pattern Recognition*, 39(5):980–987, 2006.

[54] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceeding of IEEE International Conference on Neural Networks*, pages 1942–1948, Perth,Australia, November 1995.

[55] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, February 2002.

[56] 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.

[57] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295, 2006.

[58] M. P. Wachowiak, R. Smolikova, and Zheng. Y. An approach to multimodal biomedical image registration utilizing particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):289–301, 2004.

[59] N. Franken and A. P. Engelbrechet. Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, 9(6):562–579, 2005.

[60] E. Bonabeau, M. Dorigo, and G Theraulaz. *Swarm Intelligence.* Oxford University Press, Oxford, 1999.

[61] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[62] 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.

[63] N. Meuleau and M. Dorigo. Ant colony optimization and stochastic gradient descent. *Artificial Life*, 8:103–121, 2002.

[64] G. Theraulaz and E. Bonabeau. A brief history of stigmergy. *Artificial Life*, 5:97–116, 1999.

[65] L. M. Gambardella, E. D. Taillard, and G. Agazzi. Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. In D. Corne et al., editor, *New ideas in Optimization*, pages 63–76, McGraw Hill, London, UK, 1999.

[66] K. Socha, M. Sampels, and M. Manfrin. Ant algorithm for the university course timetabling problem with regard to the state-of-the-art. In G. R. et al. Raidl, editor, *Applications of Evolutionary Computing, Proc. EvoWorkshops 2003, LNCS*, volume 2611, pages 334–345. Springer Verlag, 2003.

[67] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305, 1997.

[68] A. Shymygelska and H. H. Hoos. An ant colony optimisation algorithm for the 2d and 3d hydrophobic polar protein folding problem. *BMC Bioinformatics*, 6(30), 2005.

[69] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, pages 53–67, June 2002.

[70] S. Mishra. Hybrid least-square adaptive bacterial foraging strategy for harmonic estimation. *IEE Proceeding*, 152(3):379–389, 2005.

[71] E. Ben-Jacob, O. Shochet, A. Tenenbaum, and I. Cohen. Generic modeling of cooperative growth patterns in bacterial colonies. *Nature*, 368:46–49, 1994.

[72] C. Vlachos, R. C. Paton, J. R. Saunders, and Q. H. Wu. A rule-based approach to the modelling of bacterial ecosystems. *BioSystems*, 84:49–72, 2005.

[73] M. Ginovart, D. Lopez, and J. Valls. Indism, an individual-based discrete simulation model to study bacterial cultures. *Journal of Theoretical Biology*, 214(2):305–319, 2002.

[74] R. Gregory, R. C. Paton, J. R. Saunders, and Q. H. Wu. Parallelising a model of bacterial interaction and evolution. *BioSystems*, 76:121–131, 2004.

[75] R. Gregory, J. R. Saunders, and V. A. Saunders. The paton individual-based model legacy. *BioSystems*, 85:45–54, 2006.

[76] P. A. Spiro, J. S. Parkinson, and H. G. Othmer. A model of excitation and adaptation in bacterial chemotaxis. *Proceedings of the National Adademy of Sciences*, 94:7236–7268, 1997.

[77] R. N. Bearon and T. J. Pedley. Modelling run-and-tumble chemotaxis in a shear flow. *Bulletin of Mathematical Biology*, 62:775–791, 2000.

[78] B. A. Mello and Y. Tu. Perfect and near-perfect adaptation in a model of bacterial chemotaxis. *Biophysical Journal*, 84:2943–2956, 2003.

[79] KEGG: Kyoto Encyclopedia of Genes and Genomes. Bacterial chemotaxis - general - escherichia coli k-12 mg1655. Available Online At: http://www.genome.ad.jp/kegg/pathway/eco/eco02030.html (accessed 25 March 2006).

[80] G. H. Wadhams and J. P. Armitage. Making sense of it all: Bacterial chemotaxis. *Nature Reviews Molecular Cell Biology*, 5(12):1024–1037, 2006.

[81] L. You, R. C. Cox III, R. Weiss, and F. H. Arnold. Programmed population control by cell-cell communication and regulated killing. *Nature*, 2004.

[82] iGEM. Project x. Availabe Online At: http://parts2.mit.edu/wiki/index.php (accessed 31 January 2007).

[83] J. P. Ward, J. R. King, and A. J. Koerber. Mathematical modelling of quorum sensing in bacteria. *Journal of Mathematics Applied in Medicine and Biology*, 18:263–292, 2001.

[84] J. D. Dockery and J. P. Keener. A mathematical model for quorum sensing in *pseudomonas aeruginosa*. *Mathematical Biology*, pages 1–22, 2000.

[85] K. J. Painter and T. Hillen. Volume-filling and quorum sensing in models for chemosensitive movement. *Canadian Applied Mathematics Quarterly*, 10(4):501–543, 2002.

[86] A. B. Goryachev, D. J. Toh, K. B. Wee, and et al. Transition to quorum sensing in an agrobacterium population: A stochastic model. *PLoS Computational Biology*, pages 1–51, 2005.

[87] J. Muller, C. Kuttler, and B. A. Hense. Cell-cell communication by quorum sensing and dimension-reduction. Technical report, February 2005.

[88] J. Garcia-Ojalvo, M. B. Elowitz, and S. H. Strogatz. Modeling a synthetic multicellular clock: Repressilators coupled by quorum sensing. *Proceedings of the National Academy of Sciences*, 101(30):10955–10960, 2004.

[89] K. Grijspeerdt, J.-U. Kreft, and W. Messens. Individual-based modelling of growth and migration of *salmonella enteritidis* in hen's eggs. *International Journal of Food Microbiology*, 100:323–333, 2005.

[90] S. Cooper. What is the bacterial growth law during the division cycle? *Journal of Bacteriology*, 170:5001–5005, 1988.

[91] K. C. Chen, R. M. Ford, and P. T. Cummings. Mathematical models for motile bacterial transport in cylindrical tubes. *Journal of theoretical Biology*, 195:481–504, 1998.

[92] B. B. Youssef. Simulation of cell population dynamics using 3-d cellular automata. volume 3305, pages 561–570, Amsterdam, The Netherlands, October 2004. Springer Berlin.

[93] F. Schweitzer. *Brownian agents and active particles.* Springer - Verlag, 2002.

[94] Y. J. Cao and Q. H. Wu. Study of initial population in evolutionary programming. In *Proceedings of the European Control Conference*, volume 368, pages 1–4, 1997.

[95] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatical databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 1996.

[96] J. F. Staropoli and F. Alon. Computerized analysis of chemotaxis at different stages of bacterial growth. *Biophysical Journal*, 78:513–519, 2000.

[97] S. P. Diggle, A. S. Griffin, G. S. Campbell, and S. A. West. Cooperation and conflict in quorum-sensing bacterial populations. *Nature*, 450:411–414, November 2007.

[98] R. Paton, R. Gregory, C. Vlachos, and Q. H. Wu. Evolvable social agents for bacterial systems modeling. *IEEE Transaction on Nanobioscience*, 3:208–216, 2004.

[99] S. D. Muller, J. Marchetto, S. Airaghi, and P. Koumoutsakos. Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*, 6(1):16–29, 2002.

[100] R. C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *the 2000 Congress on Evolutionary Computation*, volume 1, pages 84–88, La Jolla, California, USA, July 2000 2000.

[101] 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, 2006.

[102] I. Borgulya. A hybrid evolutionary algorithm for the p-median problem. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 649–650, 2005.

[103] W. J. Tang, Q. H. Wu, and J. R. Saunders. A novel model for bacterial foraging in varying environments. In *Lecture Notes in Computer Science*, volume 3980, pages 556–565, Glasgow, UK, May 2006. Springer Berlin.

[104] F. van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.

[105] J. Y. Wen, Q. H. Wu, K. I. Nuttal, D. W. Shimmin, and S. J. Cheng. Construction of power system load models and network equivalence using an evolutionary computation technique. *Electrical Power and Energy Systems*, 25(4):293–299, 2003.

[106] H. A. Abbass, K. Sastryy, and D. Goldberg. Oiling the wheels of change: The role of adaptive automatic problem decomposition in nonstationary environments. Technical Report 2004029, IlliGAL Report.

[107] R. W. Morrison and K. A. De Jong. A test problem generator for non-stationary environments. In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, IEEE Press, pages 2047–2053, 1999.

[108] J. Branke. *Evolutionary optimization in dynamic environments.* Kluwer Academic Publishers, Massachusetts USA, 2002.

[109] J. Branke. Evolutionary approaches to dynamic optimization problems - updated survey. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 27–30, 2001.

[110] S. Yang. Memory-enhanced univariate marginal distribution algorithms for dynamic optimization problems. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2560–2567. IEEE Press, 2005.

[111] J. Branke. Memory-enhanced evolutionary algorithms for dynamic optimization problems. In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, volume 3, pages 1875–1882. IEEE Press, 1999.

[112] T. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In *Applications of Evolutionary Computing, Lecture Notes in Computer Science*, volume 3005, pages 489–500. Springer, 2004.

[113] J. P. Li, M. Balazs, G. Parks, and P. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3):207–234, 2002.

[114] N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In *Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, volume 1141, pages 512–522. Springer, 1996.

[115] M. Guntsch, M. Middendorf, and H. Schmeck. An ant colony optimization approach to dynamic tsp. In *Proceedings of the Genetic and Evolutionary Computation Conference 2001*, pages 860–867. Morgan Kaufmann Publishers, 2001.

[116] C. Fernandes, V. Ramos, and A. C. Rosa. Varing the population size of artificial foraging swarms on time varying landscapes. In *Lecture Notes in Computer Science*, volume 3696, pages 311–316. 2005.

[117] R. Walker. 'niche selection' and the evolution of complex behavior in a changing environment – a simulation. *Artificial Life*, 5:271–289, 1999.

[118] J. Carpentier. Contribution to the economic dispatch problem. *Bull. Soc. Franc. Elect.*, 8(3):431–447, 1962.

[119] K. Xie and Y. H. Song. Dynamic optimal power flow by interior point methods, generation, transmission and distribution. *IEE Proceedings*, 148(1):76–84, January 2001.

[120] Q. H. Wu and Y. J. Cao. Dispatching. In John G. Webster, editor, *Encyclopaedia of Electrical and Electronics Engineering.* John Wiley and Sons Inc., 1999.

[121] M. A. Abido. Optimal power flow using particle swarm optimization. *International Journal of Electrical Power and Energy Systems*, 24(7):563–571, October 2002.

[122] C. H. Lo, C. Y. Chung, D. H. M. Nguyen, and K. P. Wong. A parallel evolutionary programming based optimal power flow algorithm and its implementation. In *Proceedings of the third international conference on machine learning and cybernetics*, Shanghai, 2004.

[123] S. He, J. Y. Wen, E. Prempain, Q. H. Wu, J. Fitch, and S. Mann. An improved particle swarm optimization for optimal power flow. In *2004 International Conference on Power System Technology*, Nov. 2004.

[124] A. Homaifar, S. Y. Lai, and X. Qi. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254, 1994.

[125] 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.

[126] 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.

[127] M. S. Li, W. J. Tang, W. H. Tang, Q. H. Wu, and J. R. Saunders. Bacterial foraging algorithm with varying population for optimal power flow. In *Lectue Notes in Computer Science*, pages 32–41, 2007.

[128] S. He, E. Prempain, and Q. H. Wu. An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, 36(5):585–605, Oct. 2004.

[129] S. A. Ryder. Diagnosing Transformer Faults Using Frequency Responce Analysis. *IEEE Electrical Insulation Magazine*, 19(2):16–22, 2003.

[130] S. K. Sahoo and L. Satish. Discriminating Changes Introduced in the Model for the Winding of a Transformer Based on Measurements. *Electric Power System Research*, 2006.

[131] E. P. Dick and C. C. Erven. Transformer Diagnostic Testing by Frequency Responce Analysis. *IEEE Transactions on Power Apparatus and Systems*, 1(6):2144–2153, 1978.

[132] X. Liu and S. Qiang. Test Research on Power Transformer Winding Deformation by FRA Method. In *Proceedings of 2001 IEEE International Symposium on Electrical Insulating Materials*, pages 837–840, November 2001.

[133] F. de Leon and A. Semlyen. Efficient Calculation of Elementary Parameters of Transformers. *IEEE Transactions on Power Delivery*, 7(1):420–428, 1992.

[134] E. Rahimpour, J. Christian, and K. Feser. Transfer Function Method to Diagnose Axial Displacement and Radial Deformation of Transformer Windings. *IEEE Transactions on Power Delivery*, 18(2):493–505, 2003.

[135] E. Bjerkan and H. K. Hoidalen. High Frequency FEM-based Power Transformer Modeling: Investigation of Internal Stresses due to Network-initiated Overvoltages. *Electric Power System Research*, doi:10.1016/j.epsr.2006.08.031, 2006.

[136] W. H. Tang, Q. H. Wu, and Z. J. Richardson. A simplified Transformer Thermal Model Based on Thermal-Electric Analogy. *IEEE Transactions on Power Delivery*, 19(3):1112–1119, 2004.

[137] 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. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 3, pages 2745–2751, September 2005.

[138] A. Shintemirov, W. H. Tang, Z. Lu, and Q. H. Wu. Simplified Transformer Winding Modelling and Parameter Identification Using Particle Swarm Optimiser With Passive Congregation. In *Applications of Evolutinary Computing*, volume 4448 of *Lecture Notes in Computer Science*, pages 145–152. Springer-Verlag Berlin, 2007.

[139] V. Rashtchi, E. Rahimpour, and E.M. Rezapour. Using a Genetic Algorithm for Parameter Identification of Transformer R-L-C-M Model. *Electrical Engineering*, 88(5):417–422, 2006.

[140] N. Abeywickrama, Y. V. Serdyuk, and S. M. Gubanski. Exploring Possibilities for Characterisation of Power Transfromer Insulation by Frequency Response Analysis (FRA). *IEEE Transactions on Power Delivery*, 21(3):1375–1382, 2006.

[141] N. Abeywickrama. Modelling of high frequency response of transformers for characterization of insulation quality. Licentiate of Engineering Thesis, Chalmers University of Technology, Department of Materials and Manufacturing Technology, 2005.

[142] M. Florkowski and J. Furgal. Detection of Transformer Winding Deformations Based on the Transfer Function- Measurements and Simulations. *Measurement Science and Technology*, 14(11):1986–1992, 2003.

[143] E. Rahimpour, J. Christian, K. Feser, and H. Mohseni. Modellierung der Transformatorwicklung zur Berechnung der ubertragungsfunktion fur die Diagnose von Transformatoren. *ELEKTRIE*, 54(1):18–30, 2000.

[144] A. Shintemirov, W. J. Tang, W. H. Tang, and Q. H. Wu. Improved modelling of power transformer winding using bacterial swarming algorithm

and frequency response. *Electric Power Systems Research*, pages 1–15, 2008.

[145] W. J. Tang, Q. H. Wu, and J. R. Saunders. A Bacterial Swarming Algorithm for Global Optimization. In *Proceedings of the IEEE Congress for Evolutionary Computation (CEC 2007)*, Singapore, September 2007.