A NOVEL METAHEURISTIC ALGORITHM: DYNAMIC VIRTUAL BATS ALGORITHM FOR GLOBAL OPTIMIZATION

A THESIS SUBMITTED TO THE FACULTY OF ACHITECTURE AND ENGINEERING OF EPOKA UNIVERSITY

BY

ALI OSMAN TOPAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING

· MARCH, 2017 ·

Approval of the thesis:

A NOVEL METAHEURISTIC ALGORITHM:DYNAMIC VIRTUAL BATS ALGORITHM FOR GLOBAL OPTIMIZATION

submitted by **ALI OSMAN TOPAL** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Department of Computer Engineering, Epoka University** by,

Assoc. Prof. Dr. ..... _____
Dean, Faculty of Architecture and Engineering

Assist. Prof. Dr. ..... _____
Head of Department, **Computer Engineering, EPOKA University**

Assoc. Prof. Dr. ..... _____
Supervisor, **..... Dept., EPOKA University**

Prof. Dr. ..... _____
Co- supervisor, **..... Dept., ... University**

Examining Committee Members:

Prof. Dr. ..... _____
..... Dept. ..... University

Prof. Dr. ..... _____
..... Dept. ..... University

Assoc. Prof. Dr. ..... _____
..... Dept. ..... University

Assoc. Prof. Dr. ..... _____
..... Dept. ..... University

Date : 17.03.2017

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: .....

Signature: .....

ABSTRACT

A NOVEL METAHEURISTIC ALGORITHM: DYNAMIC VIRTUAL BATS ALGORITHM FOR GLOBAL OPTIMIZATION

Topal, Ali Osman
Ph.D., Department of Computer Engineering
Supervisor: Assist. Prof. Dr. Oguz Altun

March 2017,

A novel nature-inspired algorithm called the Dynamic Virtual Bats Algorithm (DVBA) is presented in this thesis. DVBA is inspired by a bat's ability to manipulate frequency and wavelength of the emitted sound waves when hunting. A role based search has been developed to improve the diversification and intensification capability of standard Bat Algorithm (BA). Although DVBA is inspired from bats, like BA, it is conceptually very different from BA. BA needs a huge number of population size; however, DVBA employs just two bats to handle the "exploration and exploitation" conflict which is known as a real challenge for all optimization algorithms.

Firstly, we study bat's echolocation ability and next, the most known bat-inspired algorithm and its modified versions are analyzed. The contributions of this thesis start reading and imitating bat's hunting strategies with different perspectives. In the DVBA,

there are only two bats: explorer and exploiter bat. While the explorer bat explores the search space, the exploiter bat makes an intensive search of the local with the highest probability of locating the desired target. Depending on their location, bats exchange the roles dynamically.

The performance of the DVBA is extensively evaluated on a suite of 30 bound-constrained optimization problems from Congress of Evolutionary Computation (CEC) 2014 and compared with 4 classical optimization algorithm, 4 state-of-the-art modified bat algorithms, and 5 algorithms from a special session at CEC2014. In addition, DVBA is tested on supply chain cost problem to see its performance on a complicated real world problem. The experimental results demonstrated that the proposed DVBA outperform, or is comparable to, its competitors in terms of the quality of final solution and its convergence rates.

ABSTRAKT

NJË ALGORITËM I RI METAHEURISTIK: ALGORITMI DINAMIK VIRTUAL I LAKURIQËVE TË NATËS PËR OPTIMIZIMET GLOBALE

Topal, Ali Osman
Doktoraturë, Departamenti i Inxhinierise Kompjuterike
Udhëheqësi: Assist. Prof. Dr. Oguz Altun

Mars 2017

Në këtë tezë është paraqitur një algoritëm e re e quajtur Algoritma Dinamike Virtuale e Lakuriqëve të natës (DVBA). DVBA është frymëzuar nga aftësia e lakuriqit të natës për të përdorur frekuencën dhe gjatësinë e valëve të zërit të lëshuara gjatë gjuetisë. Për të shtuar larmishmërinë dhe aftësinë e përmirësimit të standardit të Algoritmës së Lakuriqit (BA), është zhvilluar një kërkim i bazuar në role. Megjithëse DVBA është frymëzuar nga lakuriqët ashtu si dhe BA, ajo është konceptualisht shumë ndryshe nga BA. Në BA nevojitet një numër i madh popullimi; por në DVBA nevojiten vetëm dy lakuriqë për të menaxhuar konfliktin "eksploro dhe shfrytëzo", që njihet si një sfidë e vërtetë për të gjitha algoritmat e optimizimit.

Së pari studiohet aftësia e lakuriqëve për të percaktuar vendin me anë të jehonës së zërave dhe më pas analizohen algoritmat më të njohura të frymëzuara nga lakuriqët, si modelet

e modifikuara të tyre. Kontributi i kësaj teze është në studimin dhe imitimin e strategjive gjuajtëse të lakuriqëve, nga një tjetër prespektivë. Në DVBA ka vetëm dy lakuriqë: lakuriqi eksplorues dhe lakuriqi shfrytëzues. Ndërkohë që lakuriqi eksplorues kontrollon hapësirën ku do kërkojnë gjahun, shfrytëzuesi kryen një kërkim intensiv lokal me mundësinë më të lartë për të përcaktuar vendndodhjen e objektivit të kërkuar. Lakuriqët i shkëmbejnë rolet në mënyrë dinamike në bazë të vendndodhjes së tyre.

Rendimenti i DVBA është vlerësuar gjerësisht me anë të problemeve të optimizimit me kufizim 30 funksionësh të CEC 2014 dhe është krahasuar me 4 alogaritmat klasike optimale, me 4 alogaritmat më të fundit të modifikuara dhe 5 alogartima nga një seancë speciale e CEC 2014. Në vazhdimësi DVBA është testuar me anë të problemeve të kostos së zinxhirit të furnizimit për të parë rendimentin e saj në një problem të komplikuar real. Përfundimet eksperimentale treguan se DVBA e sugjeruar funksionon më mirë ose është e krahasueshme me rivalët e saj për sa i përket cilësisë së përfundimeve finale dhe shkallës së konvergjencës.

To My Family

ACKNOWLEDGEMENTS

My first words of appreciation go to my advisor, Oguz Altun, for giving me the opportunity of embarking on this doctorate and for guiding me with his teachings and knowledge all this time.
I would like to thank to my thesis progress committee members, Elton Domnori, Arban Uka, Ilir Capuni, Albana Halili, and Endri Stoja for their comments and suggestions throughout the entire thesis.
I would also like to give thanks to Yunus Emre Yildiz and Sokol Dervishi for their supports during my Ph.D. studies.
From a personal point of view, there are also many people that I would like to show appreciation for the support and help received. I could not forget Mukremin, Elton, Bledar, Arjel, Endrit and more friends since similes, coffees, journeys, ... also help me in doing this thesis.
I am thankful to my PhD thesis defense commission members for suggestions and guidance for future works. I would like to express my gratitude to Mr. Betim Cico for giving the right advice at the right time and being a source of motivation during the finalization period of the thesis.
Finally, I would like to especially thank my parents, my wife, and children for all the sacrifice they have done and for the unconditional support they have offered me in every moment. I should thank them many things so I would like to dedicate this thesis to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABA ..... Adaptive Bat Algorithm

ABC ..... Artificial Bee Colony

BA ..... Bat Algorithm

BFV ..... Best Fitness Value

CDE ..... Competitive-adaptation variant of Differential Evolution

CEC ..... Congress of Evolutionary Computation

CLSBA ..... Chaotic Local Search-based Bat Algorithm

CR ..... Crossover rate

DABA ..... Directed Artificial Bat Algorithm

DE ..... Differential Evolution

DVBA ..... Dynamic Virtual Bats Algorithm

FEs ..... Function evaluations

FWA-DM ..... Fireworks Algorithm with Differential Mutation

GA ..... Genetic Algorithm

IEEE ..... Institute of Electrical and Electronics Engineers

JADE ..... Joint Approximate Diagonalization of Eigen

LMSBA ..... Local Memory Search Bat Algorithm

LPSR ..... Linear Population Size Reduction

L-SHADE ..... Success-History based Adaptive Differential Evolution with LPSR

MC ..... Market Cost

μBA ..... Micro-bat algorithm

NABA ..... Novel Adaptive Bat Algorithm

NP ..... Number of Population

NRGA ..... Non-Uniform Real-coded Genetic Algorithm

OSM ..... Overall Success ratios of Mean

PC ..... Production Cost

PSO ..... Particle Swarm Optimization

SA ..... Simulated Annealing

SCND ..... Supply Chain Network Design

SCRM ..... Supply Cost of Raw Material

SHADE ..... Success-History based Adaptive Differential Evolution

tf ..... The number of Total Functions

TS ..... Tabu Search

WFV ..... Worst Fitness Value

CHAPTER 1

INTRODUCTION

Day by day real-life problems in such an important areas as Industry, Economy, Telecommunications, Logistics, Bio-informatics, and Commerce are becoming more complicated, competitive, and imprecise. Dimensions of problems are becoming very high, evaluating the cost of the solutions is very expensive and knowledge of the problems is vague. Getting the solution is becoming very difficult, time consuming, and expensive. In these cases, metaheuristics algorithms have become very successful to get satisfying solutions instead of *the solution*.

Metaheuristics algorithms provide an intelligent computational method that optimizes complex multi-variable optimization problems by iteratively trying to improve a candidate solution with regard to a given optimal solution.

Nature-inspired algorithms are a very important part of the metaheuristics. Many of them have been invented and improved over the past few decades and applied with success to many numerical and combinatorial optimization problems. Ant Colony Optimization [17, 19, 20, 99, 103], Particle Swarm Optimization [22, 27, 47, 102], Artificial Bees Algorithm [43, 45, 54], Evolutionary Computation [12, 35, 51], Bat Algorithm [3, 24, 106], and Artificial Immune System [41,48,66] are examples of such nature inspired algorithms. Since the real-world optimization problems are getting more complicated, higher dimensioned, and more dynamic, it seems that meta-heuristics algorithms' popularity will keep increasing.

Bat Algorithm (BA) [106] is one of the nature-inspired algorithms which was introduced

recently and has been successfully applied to solve numerous optimization problems in diverse fields. It is a population-based search algorithm that is inspired from the echolocation behavior of bats. BA, like Yang's previous algorithms, Cuckoo Search [108] and Firefly [104], combines the advantages of existing algorithms, especially Particle Swarm Optimization and Harmony search [28]. Although BA does not imitate the real bats successfully, due to its simplicity and effectiveness a large number of BA variations have been developed and applied to a wide range of real problems [2, 46, 49, 59, 78, 109]. However, as in most of the stochastic algorithms, the standard BA suffers from the premature convergence problem and it needs improvements in exploration. The random walk size and the pulse rate parameters play very important role on exploration ability of BA. So most of the researchers focused on these parameters. Recently, Local Memory Search Bat Algorithm (LMSBA) [113], Novel Adaptive Bat Algorithm (NABA) [42], Adaptive Bat Algorithm (ABA) [100], and Chaotic Local Search-based Bat Algorithm (CLSBA) are developed towards improving BA's exploration ability.

Directed Artificial Bat Algorithm (DABA) [80] is another bat-inspired algorithm proposed by Amr Rekaby in 2013. DABA differs from other versions of the Bat Algorithms in terms of how the bat's behavior is simulated. Bat is looking for a prey in its directed scope (echo waves), which has shape of a right triangle. This directed scope consists of a set of vectors and search points on these vectors. The number of vectors and visited solutions are changed during the search for prey but the distance between the visited solutions remain the same at all times. The direction of the bat does not change unless there is no better solution in its directed scope. These vectors represent the frequency and the visited locations represent the wavelength of the waves.

The main difference between DABA and BA is the interaction between bats. In BA, bats behave like the particles in PSO; however, in DABA they fly individually without any interaction with other bats. Although DABA simulated bats behavior better than BA in nature, because of the unconnected behavior of the bats in DABA, BA is more applicable and useful.

Although BA and DABA are inspired from echolocation behavior of bat's hunting strategies, the virtual bats, in the algorithms, do not imitate the real bats successfully.

In this thesis, another bat inspired algorithm is proposed: Dynamic Virtual Bats Algorithm (DVBA) [92] [93]. DVBA is not a BA variation, it is a new simulation of the bat's hunting strategies, in which bat's echolocation behavior is imitated completely. In DVBA, a role-based search is developed to improve the diversification and intensification capability of the Bat Algorithm. There are only two bats: explorer and exploiter bat. While the explorer bat explores the search space, the exploiter bat makes an intensive search of the local with the highest probability of locating the desired target. During the search bats exchange the roles according to their positions. Experimental results show that DVBA reaches accurately and reliably the global optimum better than compared algorithms. It represents a particular way of coping with the "exploration and exploitation" conflict.

In this dissertation we focus on the next objectives:

1. Make an in-depth study of echolocation and bat algorithm.
2. Develop a new algorithm by imitating the bat's hunting strategies that can manage the exploration and exploitation conflict in an effective and efficient way.
3. Validate the performance of the proposed algorithm DVBA by comparing with classic and state-of-the-art optimization algorithms.
4. Analyze the performance of dynamic virtual bats algorithm on a real world problem.
5. Study on possible improvements for dynamic virtual bats algorithm by using the advantages of the bat algorithm and analyze it.

This thesis has been structured according to these objectives in 6 chapters that are described below. Chapter 2 provides the background needed to understand the research area. It gives the basic concepts of metaheuristics. Then, some of the well-known metaheuristics algorithms are briefly described.

Since, the proposed algorithm is inspired from bats, bat's behavior, especially the way bats use echolocation are introduced in Chapter 3. Then, other bat inspired algorithms are examined with details to show differences with DVBA.

Chapter 4 describes the proposed algorithm DVBA. The motivation of DVBA and how bats are simulated in the algorithm are presented. Then, the effects of the parameters on the performance of DVBA are analyzed.

In Chapter 5, the performance of the DVBA is extensively evaluated on 6 classical test functions and 30 CEC 2014 benchmark functions and compared within three groups of algorithms. In group 1, standard Bat Algorithm, original Particle Swarm Optimization, Genetic Algorithm, and Simulated Annealing are used. In group 2, four state-of-the-art modified BA algorithms are compared and group 3, five algorithms from a special session at CEC 2014 are used in comparison, as well. In addition, DVBA is applied to minimize the supply chain cost with other well-known algorithms. The analysis done to address the questions:

Given a set of optimization test function, is DVBA better to solve them comparing to other algorithms in terms of accuracy.

Can DVBA keep up the same achievement for different type of optimization problems?

How does the performance of DVBA change when the dimension of the problem increases?

In Chapter 6, a hybrid of DVBA and Bat Algorithm is presented Micro Bat Algorithm (μBA). The advantages of DVBA and BA is combined in this hybrid and it is compared with the original BA and DVBA.

Chapter 7 is devoted to discuss the global conclusion of this dissertation as well as the future lines of research. The bibliography is presented at the end of the dissertation.

CHAPTER 2

METAHEURISTICS

Optimization algorithms can be divided in two categories: deterministic and stochastic algorithms. Deterministic algorithms are able to find the optimal solution, whereas stochastic algorithms cannot. Deterministic algorithms typically constitutes a brute-force style approach which is suitable for very small problem instances. In the case of large-scale nonlinear global optimization problems, brute-force approach will be extremely time-consuming process. On the other hand, while stochastic algorithms do not always produce optimal solutions, they do operate in polynomial time and might be able to produce solutions that are 'good enough' for practical purposes [52, 69].

Metaheuristics is a major or the primary subfield of stochastic algorithms. In general, there are two types for stochastic algorithms: heuristic and metaheuristics. Both employ some degree of randomness to find optimal (or near-optimal) solutions to hard problems. Specially, metaheurisitcs are the most general method which are applied a very wide range of problems [61, 105]. As for the term metaheuristic, it was firstly used by Glover in 1986 [29]. In metaheuristic, *meta* means 'beyond' or 'higher level', and they generally perform better than simple heuristics. *Heuristic* means 'to find' or 'to discover by trial and error'. There is no guarantee that these algorithms will find the optimal solutions, but they can be an efficient way to produce acceptable solutions in a reasonably practical time.

There is no commonly accepted definition for the term metaheuristic. According to Voss, Martello, Osman and Roucairol [98], "A metaheuristic is an iterative master process that

guides and modifies the operations of subordinate heuristics to efficiently produce high-quality solutions. It may manipulate a complete (or incomplete) single solution or a collection of solutions at each iteration. The subordinate heuristics may be high (or low) level procedures, or a simple local search, or just a construction method."

Osman and Laporte's definition [68] for metaheuristic is "A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions."

Metaheuristics algorithms have two major components: exploration (diversification) and exploitation (intensification). Exploration can be described as the ability to test various regions in the search space. Exploitation is the ability to focus the search around a determined solution in order to locate the optimum solution precisely [53]. Both components are working together to discover the best solution in the search area. There is, however, a trade-off between exploration and exploitation, so that decreasing either will only serve to increase the other. If an algorithm is more exploratory and less exploitative, it's convergence speed will slow down and thus decrease the accuracy. Conversely, if the algorithm is more exploitative and less exploratory, it is more likely to be trapped at local optima, which makes it very difficult to find the global optimum. In order to achieve effective performance on problem optimization, these two components should be well balanced [12]. At the end, *how accurately* and *how quickly* a metaheuristic algorithm is able to produce solutions will usually be the criteria to judge how effective it actually is.

Metaheuristic algorithms can be divided into two categories namely, nature inspired and evolutionary. Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Bat Algorithm (BA), Bacterial Foraging Optimization, Simulated Annealing, and Artificial Bee Colony (ABC) are examples of such nature inspired algorithms. Evolutionary algorithms are Genetic Algorithm (GA), Differential Evolution (DE), and Evolutionary Strategies (ES).

In the rest of this chapter, some of the most known metaheuristic algorithms are summarized, including Simulated Annealing (SA), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Tabu Search (TS), Differential Evolution (DE), and Artificial Bee Colony (ABC). And

also some of the improved versions of these algorithms are mentioned, including FWA-DM, L-SHADE, b6e6rl, NRGA, and OPTBees.

2.1 Simulating Annealing

Simulated Annealing [1] [60] algorithm simulates the annealing process. The algorithm decreases getting stuck in local optima by occasionally accepting solutions that are worse than the current solution. This version with random restarts, re-starts search from a random position after local search fails to improve a given time (limit l). As seen in Figure 6, the algorithm starts with initializing (Node 2) constants limit l, initial temperature t_0, and cooling scheduler c. If there is time to another annealing (Node 3), algorithm resets (Node 4) current temperature with $t \leftarrow t_0$, number of bad trials with $n \leftarrow 0$, and current solution with $s \leftarrow$ random solution. If number of bad trials n is less than limit l, we continue (Node 5) with improving current solution by producing a tweak of it (Node 6). If the tweak r is better than s (Node 7-8), we directly update current solution: $s \leftarrow r$. The interesting thing about Simulated Annealing is that we still update when r is worse than s if r is lucky, e.g. if (2.1) holds, where $rand(0, 1)$ is a random value between 0 and 1, and $|.|$ denotes absolute value.

$$rand(0, 1) \leq exp(-\frac{|f(s) - f(r)|}{t}) \tag{2.1}$$

This way the algorithm has a chance of escaping local optima. When there is an update, the number of bad trials is reset: $n \leftarrow 0$. When an update was not done, the number of bat trials is incremented: $n \leftarrow n + 1$ (Node 9). In each iteration the temperature t is cooled down (Node 10) using a schedule constant c that has a value less than $1 : t \leftarrow ct$.

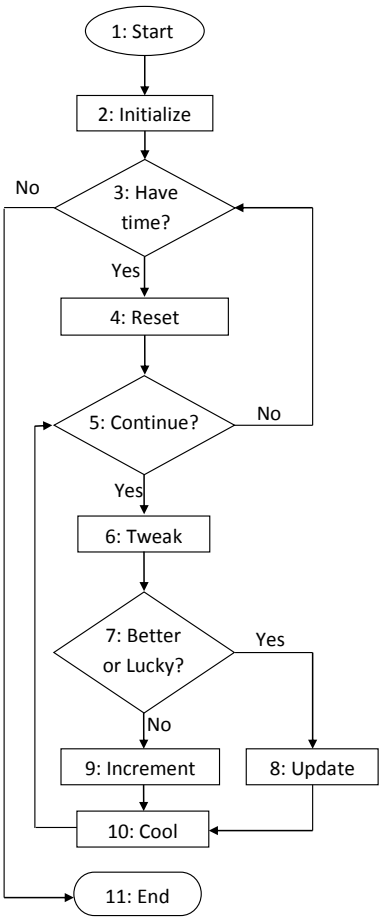


Figure 2.1: Simulated Annealing with Random Restarts flowchart [60].

2.2 Genetic Algorithm

Genetic algorithm (GA) was proposed by John Holland in the 1970s [35]. Genetic Algorithm is very important part of the evolutionary algorithms which is inspired from natural evolution. It iterates through fitness assessment, selection, reproduction, mutation, and recombination [31, 65].

As seen in Fig. 2.2, GA starts with initialization of population X (Node 2) randomly. If there is time to generate another generation of solutions (Node 3), the algorithm proceeds to

produce a new generation. E.g. if still a new child is needed to complete the new generation (Node 4), two new children are generated in Node 5. In this step (Node 5), first two new parents are selected. From the two parents, two children are produced by crossover. Each child is mutated, and then added to the list of new generation solutions. In Node 6, the old generation is completely overwritten with the new generation. If no new generation is needed, the algorithm ends (Node 7).

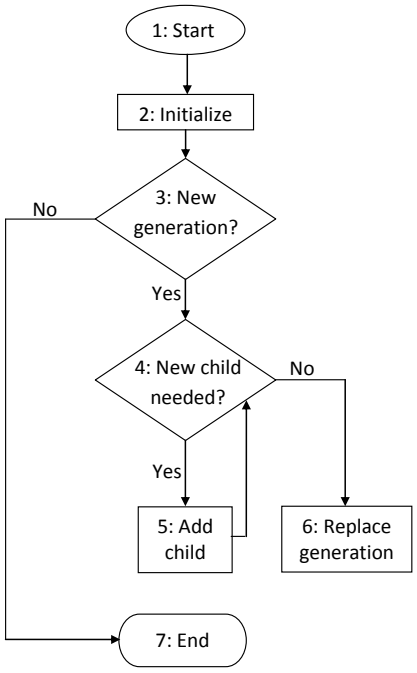


Figure 2.2: Genetic Algorithm flowchart [35].

2.3 Particle Swarm Optimization

Particle Swarm Optimization is a population based stochastic optimization method developed by Eberhart and Kennedy [47] in 1995. The algorithm, which is based on a metaphor of social interaction, searches a space by adjusting the trajectories of individual vectors, called particles as they are conceptualized as moving points in multidimensional space. The individual particles

are drawn scholastically toward the positions of their own previous best performance and the best previous performance of their neighbors [11].

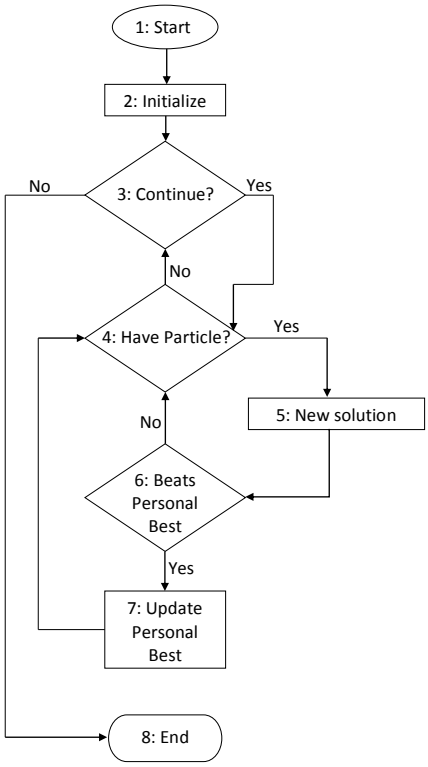


Figure 2.3: Particle Swarm Optimization flowchart. [47]

PSO algorithm is depicted in Fig. 2.3. The algorithm starts with Node 1. In Node 2, we initialize particle positions x_i and particle velocities v_i, randomly. In addition particle personal best positions are initialized as equal to starting positions: $p_i = x_i$, and the global best position $x_* = x_{argmax}(f(p_i))$. In Node 3 if we decide to continue, because e.g. we have more time, we start processing each particle i in the next generation t one by one. While we have unprocessed

particles (Node 4), we get a new solution (Node 5) using (2.2) and (2.3)

$$v_i(t+1) = wv_i(t) + c_1 r_1(p_i - x_i(t)) + c_2 r_2(x_* - x_i(t))) \tag{2.2}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.3}$$

where w, c_1, and c_2 are constants and r_1 and r_2 are random values between $[0, 1]$ drawn from the uniform distribution. If the new solution is better than personal best ($f(x_i(t+1)) > f(p_i)$) as in Node 6, we update personal best in Node 7: $p_i = x_i(t+1)$. When we have no more particles to process (Node 4), we get out of particle loop. When the termination criteria is met (e.g. "No" edge in Node 3) the algorithm ends.

2.4 Tabu Search

Tabu Search [60] algorithm keeps a list of solutions that are already evaluated (tabu), and avoids reevaluating those solutions. As depicted in Fig. 2.4 the algorithm starts by initializing (Node 2) an empty tabu list L, getting a random current solution s, and adding s to L. While there is enough time (Node 3), we proceed to building a new solution r by tweaking the existing solution s (Node 4). If r is not in the tabu list (Node 5), current solution is updated (Node 5): $s \leftarrow r$, and r is added to the tabu list L.

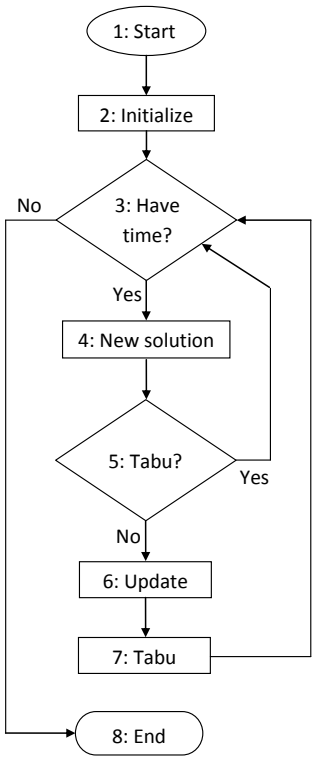


Figure 2.4: Tabu Search flowchart [60].

2.5 Differential Evolution (DE)

Differential Evolution [71] [85] is another algorithm that simulates the evolutionary behavior. As seen in Fig. 2.5, the algorithm starts with initializing the population of individuals X randomly. While we have time to work on another generation (Node 3), the algorithm takes a copy of the current population into parents Q (Node 4), and iterates through each parent q (Node 5). In each iteration a new child e is made through (2.4) and (2.5), where a, b, and c are random parents that are different from q and each other, m is the constant $mutationrate$, and $\otimes$ represents the genetic crossover operator.

$$d = a + m(b - c) \tag{2.4}$$

$$e = q \otimes d \tag{2.5}$$

If the child e is better than the parent q, e replaces corresponding element in X.

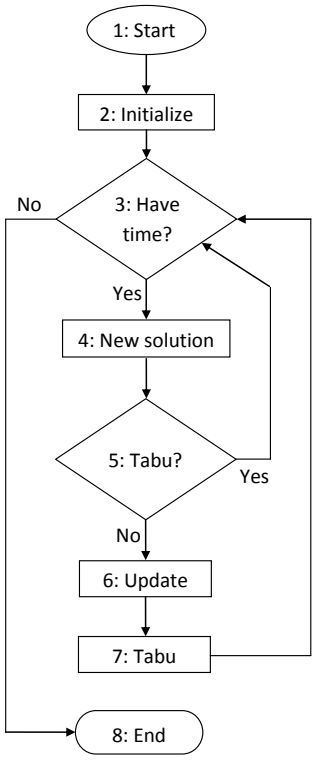


Figure 2.5: Differential Evolution flowchart derived from [71].

2.6 Artificial Bee Colony (ABC)

Articial Bee Colony (ABC) [43] [45] algorithm simulates behavior of bees in a bee hive in search of food sources. As depicted in Fig. 2.6, the algorithm starts with initializing random initial

solutions X (in ABC metaphor each solution is a food source).

If there is enough time (Node 3), the algorithm process to updating food sources. In each iteration, each food source is "visited" once (Node 4). Visiting a solution x_i entails making a recombination of it with a random other solution x_j as in (2.6), where d is a random dimension, and r is a uniform random number in the range [-1, 1].

$$x_i[d] \leftarrow x_i[d] + r(x_i[d] - x_i[d]) \tag{2.6}$$

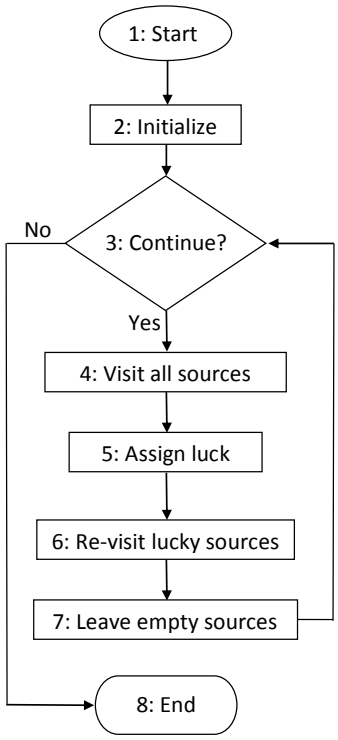


Figure 2.6: Artificial Bee Colony flowchart derived from [43].

In Node 5 each food source is assigned a probability of being re-visited, e.g. "luck", in the same iteration, based on the value of f. The food source with better f value has a higher

probability P of being re-visited (2.7):

$$P_i = \begin{cases} f(x_i) + 1, & \text{if } f(x_i) \geq 0. \\ \frac{1}{1-f(x_i)}, & \text{otherwise.} \end{cases} \tag{2.7}$$

In Node 6 if the food source is lucky, e.g. if $P_i > rand(0, 1)$, where $rand(0, 1)$ is a random number in the range [0, 1] from the uniform distribution, the food source is re-visited using (2.6). This ensures that the neighborhood of better solutions are visited more, hence makes the algorithm more elitist/exploitist. In Node 7 the algorithm checks whether any of the current solution neighborhoods failed to produce any improvement for the last *limit* iterations. Such neighborhoods are abandoned for a random new neighborhood.

CHAPTER 3

BAT'S ECHOLOCATION AND BAT ALGORITHMS

In this section, bat's hunting strategies, bat algorithm, and modified bat algorithms are examined.

3.1 Bats and Echolocation

Bats have the unique ability to detect insects and avoid obstacles around themselves by using echolocation calls [23, 83] that are usually ultrasonic (ranging in frequency from 20 to 200 kilohertz (kHz)). Bats emit sound waves and listen to the returning echoes. From these, bats can generate a 3D blueprint of their environment. Bats can distinguish the shape, size, and texture of a tiny prey, in which direction the prey is heading, and even the speed of the prey by using the delayed time and loudness of the response [33, 67]. If prey moves towards the bat, the returning echo will have a higher pitch than the original sound, while the echo from prey moving away the bat will have a lower pitch. This difference is due to the Doppler Effect [37].

Bats have also the ability to change the way they emit the sound pulses. By varying frequency of the pulse, bats can change the traveling range of the pulses. Frequency f is inversely proportional with the wavelength λ (Fig.3.1) and multiplication of these gives us the speed of sound as in (3.1) where $V = 343$ m/s in air.

$$V = f\lambda \tag{3.1}$$

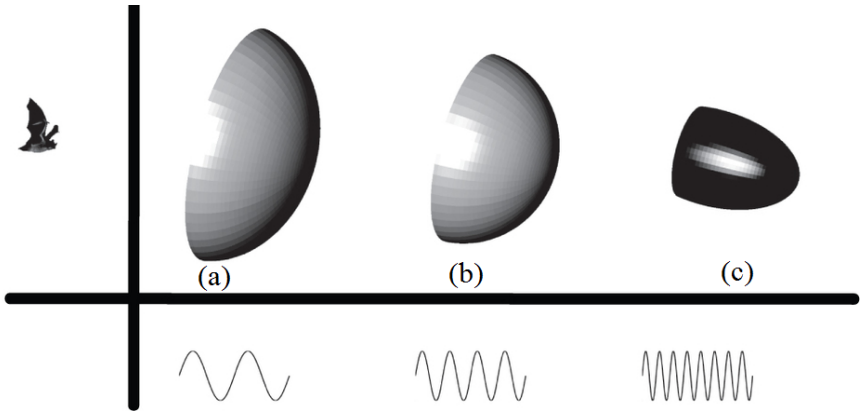


Figure 3.1: Emitted sound waves width decreases as frequency increases that is decrease in wavelength. [38]

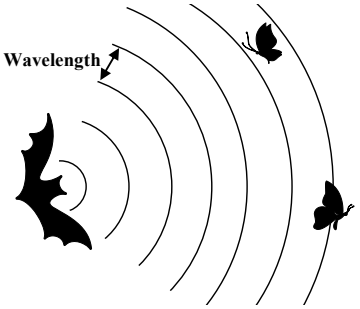
Figure 3.1 illustrates that for constant energy, an increase in frequency (left to right), that is decrease in wavelength. [38] When bats hunt, they burst sound pulses with lower frequency and longer wavelength, hence the sound pulses can travel farther distance. In this long range mode it becomes hard to detect the exact position of the prey (Fig.3.2a), but it becomes easy to search large area [37].

When bats detect prey, the pulses will be emitted with higher frequency and shorter wavelength, so that bats are able to update the prey location more often (Fig.3.2b). Depending on the species, the range of the sound pulses could be from 2.4m to 62m [4, 84].

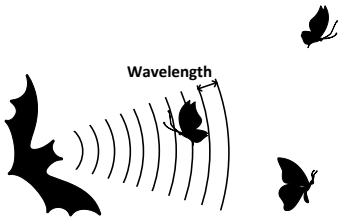
Besides frequency and wavelength, bats also change the loudness (intensity) of the sound. Bats emit sound as low as 50dB and as high as 120 dB [87] [88], that is enough to damage human hearing. Since this sound waves has ultrasonic frequency, we are unable to hear it. Emitted sound is louder than which varies from loudest when searching for prey to a quieter base when approaching to prey; therefore, we can say that loudness and frequency of sound pulses (rate of pulse) are inversely proportional.

To summarize, there are two conspicuous properties of bat hunting strategy:

1. When a bat searches for prey, its sound frequency is lower and wavelength is longer and with high-intensity (very loud) (Exploration)(Fig.3.2a).



(a) Decreases frequency that is increase in wavelength



(b) Increases frequency that is decrease in wavelength

Figure 3.2: Two conspicuous properties of bat hunting strategy [83]

2. When a bat detects prey, its sound frequency is higher and wavelength is shorter and with lower intensity (quieter) (Exploitation)(Fig.3.2b).

It is clear that the problem of exploration and exploitation balance has naturally been resolved in the case of bats.

3.2 Bat Algorithm

Bat Algorithm was introduced by Yang [106] as a new meta-heuristic method that was based on the echolocation behavior of bats. It is more like a combination of Particle Swarm Optimization

and Harmony Search [106]. Although BA does not imitate the real bats successfully, due to its simplicity and effectiveness a large number of BA variations have been developed and applied to a wide range of real problems [59, 78, 109, 113].

Bat algorithm [106] is governed by three idealized rules: 1) Bats can detect the distance between prey, food, and the obstacles by using echolocation. 2) During the search for prey, bats fly randomly with velocity V_i, with fixed sound pulse frequency f_i, varying wavelength λ, and loudness A_0. 3) Loudness can change from large value A_0 to minimum constant value A_{min}. Besides these rules, bat algorithm also assumes that the frequency f varies in a range $[f_{min}, f_{max}]$.

Initially, ith bats' position x_i, rate of pulse r_i, loudness A_i, pulse frequency f_i, and velocity V_i are determined randomly. In the main loop the position x_i^t and the velocity V_i^t of the bats each time step t are updated as in (3.2), (3.3), and (3.4) [106],

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{3.2}$$

$$V_i^t = V^{t-1} + (x_i^t - x_*)f_i, \tag{3.3}$$

$$x_i^t = x_i^{t-1} + V_i^t, \tag{3.4}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution and x_* is the current global best position at time step t.

For the local search phase, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$x_{new} = x_{old} + \epsilon A^t, \tag{3.5}$$

where $\epsilon \in [-1, 1]$ is a random number, $A^t =< A_i^t >= \frac{1}{N}\sum_{i=1}^{N} A_i^t$ is the average loudness of all the bats at this time step, and N is the number of bats.

Loudness A_i and the rate of pulse r_i change during the iteration process. When a bat gets closer to prey its loudness decreases and the rate of pulses increases. In algorithm these changes are shown as follows:

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - exp(-\gamma t)], \tag{3.6}$$

where α and γ are constants. In the simplicity case, α and γ are set to 0.9. According to all these approximations and idealizations bat algorithm can be given as in Algorithm 1.

Algorithm 1 BAT algorithm pseudo code

```
Objective function f(x), x = (x_1, ..., x_d)^T
Initialize the bat population x_i, v_i (i = 1, 2, ..., n)
Define pulse frequency (f_i) at x_i
Initialize pulse rates r_i and the loudness A_i
while (t < Max number of iterations) do
    Generate new solutions by adjusting frequency,
    and updating velocities and locations/solutions
    [equations (3.2) to (3.4)]
    if (rand > r_i) then
        Select a solution among the best solutions
        Generated a local solution around the selected best solution
    end if
    Generate a new solution by flying randomly
    if (rand < A_i  &  f(x_i) < f(x_*)) then
        Accept the new solutions
        Increase r_i and reduce A_i
    end if
    Rank the bats and find the current best x_*
end while
Post process results and visualization
```

In the main loop, bats will move from their current positions towards global best bat's position. While they fly towards to the global best, if any bat finds a better position, the bat's rate of pulse and loudness will change and the global best bat will be updated. This process will be repeated until the termination criteria met.

3.3 Bat Related Algorithms

As in most of the stochastic algorithms, the standard BA suffers from the premature convergence problem and it needs improvements in exploration. The random walk size and the pulse rate parameters play very important role on exploration ability of BA. As a result most of the researchers focused on these parameters. Recently, Local Memory Search Bat Algorithm (LMSBA) [113], Novel Adaptive Bat Algorithm (NABA) [42], Adaptive Bat Algorithm (ABA) [100], and Chaotic Local Search-based Bat Algorithm (CLSBA) [9] are developed towards improving BA's exploration ability. In this section, these algorithms will be examined.

3.3.1 Novel Adaptive Bat Algorithm (NABA)

NABA is proposed by Kabir [42] to improve the explorative characteristics of BA. The NABA incorporates two techniques within BA, which include the Rechenberg's 1/5 mutation rule and the Gaussian/Normal probability distribution to produce mutation step size.

NABA, like LMSBA, offers new equation to generate new solution rather than doing just random walk. It modifies the random walk equation (Eq.3.5) in line 11 of original Bat algorithm. NABA controls the random walk step size by the variance of Gaussian/Normal distribution. The modified equation is as follows:

$$x_{new} = x_{old} + \epsilon A^t N(0, \sigma) \tag{3.7}$$

where σ is the standard deviation.

Kabir used the Rechenberg's 1/5 mutation rule [79] to adaptively change the random walk step size and pulse rate to control the exploration and exploitation. According to the Rechenberg's mutation rule, the ratio of successful mutations to all mutations should be 1/5. Changing the pulse rate and standard deviation according to 1/5 rule in every m number of

cycles is performed as in following equations: Pulse rate r,

$$r(t+1) = \begin{cases} r(t) * 0.85, & \text{if successrate}(m < 1/5). \\ r(t)/0.85, & \text{if successrate}(m > 1/5). \\ r(t), & \text{otherwise} \end{cases} \tag{3.8}$$

Standard deviation σ

$$\sigma(t+1) = \begin{cases} \sigma(t) - 0.0001, & \text{if successrate}(m < 1/5). \\ \sigma(t) + 0.0001, & \text{if successrate}(m > 1/5). \\ \sigma(t), & \text{otherwise} \end{cases} \tag{3.9}$$

From the equation (3.8) and (3.9), we can see that if the success-rate is less than 1/5, NABA is exploring too much and it intends to move the bats near to the best solution and decreases the step size. If the success-rate is more than 1/5, NABA is exploiting local optima too much and it increases the step size. That will help bats to reach the local optima faster, but the accuracy will decrease because of the long step size.

3.3.2 Local Memory Search Bat Algorithm (LMSBA)

LMSBA was introduced by Yuanbin [113]. He said that if the information that has been found out in the process of the implementation was fully combined and applied in BA, it would improve BA. So LMSBA, same as PSO, keeps the bat's best solutions during the search process and uses them to generate an alternative local solution to the random walk (Eq.3.5) in BA. The alternative solution is calculated as follows:

$$x_{new2} = x_{old} + c.rand.(x_{*d} - x_{old}) \tag{3.10}$$

where x_{*d} is the current best that i-bat has found.

In LMSBA, x_{new1} (Eq.3.5) and x_{new2} (Eq.3.10) is compared and the better one is used as

x_{new}. By using this method, LMSBA intends to create local extreme search in BA local search. However, this method alone might lead the bats to a local optima and they can be trapped there easily. Because, once they fly to the vicinity of the best solution, their personal best will be same as the best solution and the alternative solution will not help them to escape from local optima trap.

3.3.3 Adaptive Bat Algorithm (ABA)

Wang presented an improved bat algorithm [100] to solve BA's premature convergence problem. Same as NABA and LMSBA, he improved the random walk equation, but he also modified the frequency and the velocity equations in BA. In BA, each bat uses the same frequency increment for the velocity which makes bat's flight behavior lack of flexibility. In ABA, Wang proposed a new method to let each bat dynamic and adaptively adjust its flight speed and its flight direction. Instead of the equations (3.2), (3.3), and (3.4) in BA, he designed the new equations as follows:

$$f_{ij} = f_{min} + (f_{max} - f_{min})\beta_{ij}, \tag{3.11}$$

$$w_{ij}^{t} = w_0(1 - exp(-\mu|x_{ij}^{t-1} - x_{*}^{t-1}j|)) \tag{3.12}$$

$$v_{ij}^{i} = w_{ij}^{t}.v^{t} - 1_{ij} + (x_{ij}^{t-1} - x_{*j}^{t-1})f_{ij} \tag{3.13}$$

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}, j = 1, ..., n \tag{3.14}$$

where μ is a positive constant, w_0 is a positive constant and $1 \leqslant w_0$. ABA is targeting to increase the speed of the bat which is farther from the prey. The farther the distance between the bat and its prey (global best solution), the faster the speed flying to its prey.

Secondly, Wang improved the random walk by combining it with shrinking search. Bats

which are far away from the current best, make random fly near to the current best. The step size of random fly within the range $[-1, 1]$ in BA while it shrinks in ABA as the iteration proceeds. ABA targets to increase the intensification of the search by shrinking search method.

3.3.4 Chaotic Local Search-based Bat Algorithm (CLSBA)

CLSBA [9] is another modified version of Bat Algorithm. It is a combination of the standard BA with chaotic sequences generated by the logistic map. The use of chaos makes the frequency adaptive and more random in nature to balance the trade-off between exploration and exploitation. The frequency of pulse emission is modified as follows in CLSBA:

$$f_i(t+1) = \mu \times f_i(t) \times (1 - f_i(t)) \tag{3.15}$$

While the step size of random walk parameters ϵ (Eq.3.5) varies within the range $[-1, 1]$ in BA, it varies within the range $[-SF(t), SF(t)]$ in CLSBA. SF is the scaling factor which changes dynamically according to the Rechenberg's mutation rule. Having a lower value of SF(t) increase the exploitation, a larger value of SF(t) accelerates the exploration. Changing the scale factor according to 1/5 rule in every m number of cycles is performed as in following equation:

$$SF(t+1) = \begin{cases} SF(t) * 0.85, & \text{if successrate} < 1/5 \ . \\ SF(t)/0.85, & \text{if successrate} > 1/5 \ . \\ SF(t), & \text{otherwise} \end{cases} \tag{3.16}$$

CHAPTER 4

DYNAMIC VIRTUAL BATS ALGORITHM

The proposed Dynamic Virtual Bats Algorithm (DVBA), like Bat Algorithm (BA), is fundamentally inspired by bat's hunting strategies, but it is conceptually very different from BA. It is a new simulation of the bat's hunting strategies. In DVBA, a role based search is developed to avoid deficiencies of BA. Although DVBA is a population based meta-heuristic, it needs just two bats to find the optimal solution. These bats are called explorer bat and exploiter bat. While the explorer bat explores the search space the exploiter bat makes a concentrated search around the best found solutions by using echolocation. Experimental results demonstrate that DVBA is more accurate and reliable at attaining the global optimum when compared with existing algorithms. DVBA represents a particular way of coping with the "exploration and exploitation" conflict.

In this chapter, all the details about dynamic virtual bats algorithm (DVBA) will be explained under three subsections.

4.1 Mathematical representation of search scope of the virtual bat

In this work we focus on how bats alter the sound frequency and the wavelength during the search. While a bat is looking for prey, there are two prominent behaviours:

1. Exploration: the bat emits sound waves with low frequency and long wavelength. Waves

are like a light bulb, illuminating a wide circle (Fig.3.1)a. This behavior is simulated as in Fig.4.1a

2. Exploitation: when the bat gets closer to the prey, it increases the frequency and decreases the wavelength of the waves to get the exact location of the prey. Waves are like a flashlight, spreading narrow beams [39] (Fig.3.1c). This behovior is simulated as in Fig.4.1b

This hunting strategy inspired us to develop DVBA. DVBA does not need to have a huge number of population size to provide the optimal solution. It is able to manage the classic exploration and exploitation trade off problem successfully by using just two bats. Each bat has its own role in the algorithm and during the search they exchange these roles according to their positions. We called these bats the explorer bat and the exploiter bat. The bat which is in a better position becomes the exploiter; meanwhile the other one becomes the explorer. While the exploiter bat increases the intensification of the search around preferable position, the explorer bat will keep looking for a better position. Until the explorer bat finds a better position, the exploiter bat will increase intensification of the search after each iteration to attain the optimal solution.

In Fig.4.1a and Fig.4.1b the triangle (▲) represents the bat and the plus (+) represents the prey. The black dots are the positions (H) on the waves which are going to be checked for a better solution.

As shown in Fig.4.1a during exploration the search points which are created by the explorer bat are distributed widely in the search space. However in Fig.4.1b the exploiter bat created a very small search scope where search points have become closer to each other.

Similar to real bats, virtual bats are looking for superior solutions (prey) in their search scope. They start flying from random positions X_i with random velocities V_i using default search scope range. During the search, the range of the search scope changes dynamically: it expands during the search (Fig.4.1a) or it shrinks if prey gets closer (Fig.4.1b). The length and the width of the search scope is controlled by wavelength λ_i and the frequency

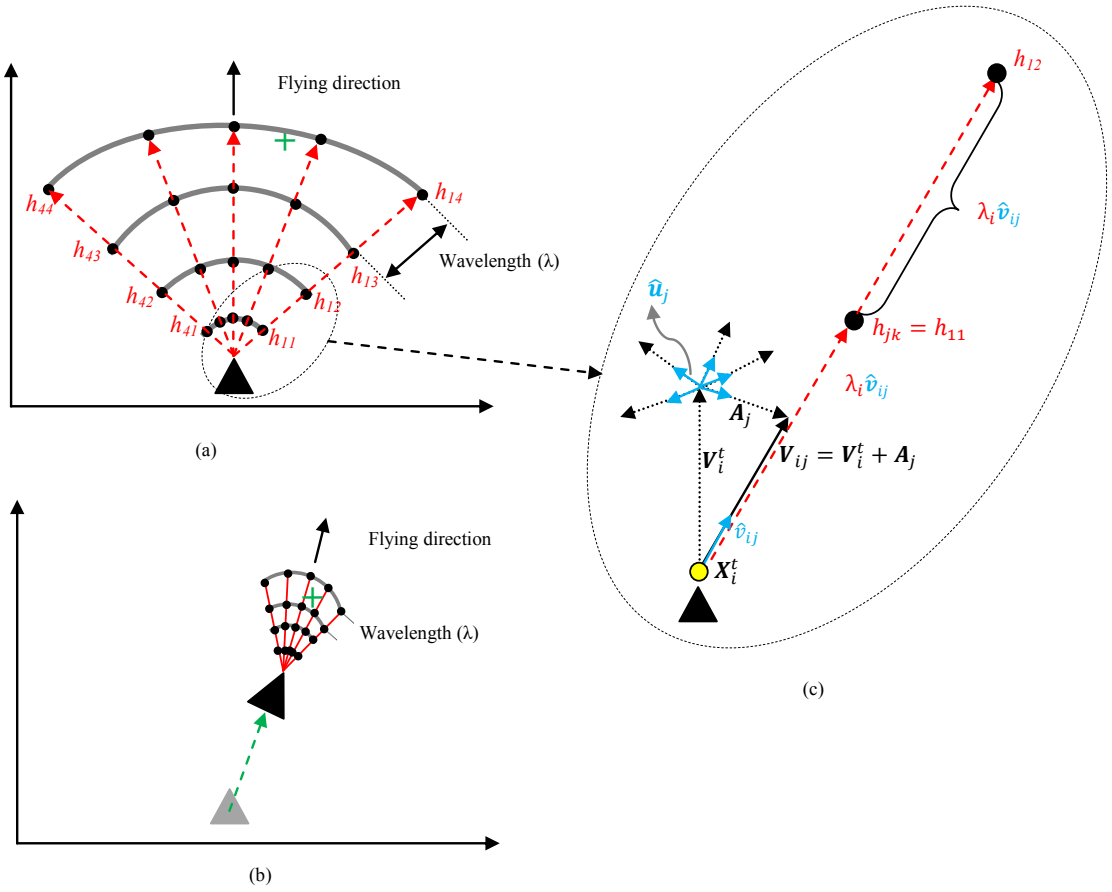


Figure 4.1: (a) Exploration: Explorer bat searching a prey, (b) Exploitation: Exploiter bat chasing a prey, (c) Search positions in the search scope of a bat.((a) zoomed in by x6)

f_i, respectively. The frequency f_i and the wavelength λ_i are used as scalar dimensionless quantity in equations(e.g. 50Hz is accepted as 50 dimensionless quantity).

The search scope is simulated by using wave direction vectors V_j^i and search points h_{jk} on the vectors as shown in Fig.4.1a and Fig.4.1c. To distribute the wave direction vectors in the search scope we have generated the unit vectors $\hat{u}_j$ which give direction to the scope width vectors $\mathbf{A}_j$ (Fig.4.1c).The unit vectors $\hat{u}_j$ are generated randomly; consequently, the wave vectors are distributed randomly in the search scope as well. The magnitude of A_j changes the width of the search scope, and is inversely proportional with the frequency of the waves f_i (7). When the frequency f increases reflect what is shown in Fig.4.1b. The search points h_{jk} are distributed by scaling the unit vector $\hat{v}_{ij}$ (9) with the same length as wavelength λ_i. The scope width vector $\mathbf{A}_j$, the wave vector $\mathbf{V}_{ij}$, and the positions of the search points h_{ij} in the bat's search scope at time step t are given by (4.1 - 4.5),

$$\mathbf{A}_j = \frac{\hat{u}_j b}{f_i}, (j = 1, ..., w), (i = 1, ..., n) \tag{4.1}$$

$$\mathbf{V}_{ij} = \mathbf{V}_i^t + \mathbf{A}_j \tag{4.2}$$

$$\hat{\mathbf{v}}_{ij} = \frac{\mathbf{V}_{ij}}{\|\mathbf{V}_{ij}\|} \tag{4.3}$$

$$h_{j,k} = \mathbf{X}_i^t + \hat{\mathbf{v}}_{ij} k \lambda_i, (k = 1, ..., m) \tag{4.4}$$

$$\mathbf{H}_{j,k} = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,k} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ h_{j,1} & h_{j,2} & \cdots & h_{j,k} \end{bmatrix} \tag{4.5}$$

where w is the number of wave vectors, i is the index of bat (maximum 2), $b \in (20, 40)$ is the

scope width variable, and m is the number of search positions on the wave vectors. Increasing the number of m and w will offer a more detailed search, but it will require longer computation time. For the problems in Table 1, we set $m = 5$ and $w = 6$.

If the frequency increases, vectors will get closer and the wavelength (distance between search points) will get shorter (4.1). This is inspired from a bat's dynamic search ability for hunting. The changes of frequencyf_i, and wavelength λ_i at time step $t + 1$ are given by (4.6 - 4.8),

$$f_i^{t+1} = f_i^t \pm \rho \tag{4.6}$$

$$\lambda_i^{t+1} = \lambda_i^t \pm \rho \tag{4.7}$$

$$\rho = mean(\frac{\mathbf{U} - \mathbf{L}}{\beta}), \qquad \{\beta \in \Re : \beta > 0\} \tag{4.8}$$

where ρ and β are positive real constants, U is the upper bounds and L is the lower bounds of the search space, and ρ is used as increment rate for frequency and wavelength. β is used as increment rate divisor (4.6), (4.7). In our simulation, we set $\beta = 100$.

Increasing β might cause bats become trapped in a local minimum for multimodal problems, but it will increase accuracy in unimodal problems. Choosing the range of the wavelength is very important. If the wavelength is too short, in large search spaces, convergence could be very slow. Conversely, if the wavelength is too long, it might bypass the global best and fail to locate the optimum position. To overcome this problem the length of the wavelength is linked to the range of the problem. For simplicity the following approximations were used. The wavelength range λ, in a range $[\lambda_{min}, \lambda_{max}] = [\rho, 5\rho]$ corresponds to a range of the frequency $[f_{min}, f_{max}] = [\rho, 5\rho]$.

In Fig.4.2 we tested DVBA with the various values of β to show its effect on the convergence characteristics of the algorithm. When β is increased, DVBA performance gets worse. This is because it decreases the increment rate ρ (4.8); therefore, the search scopes of both bats get

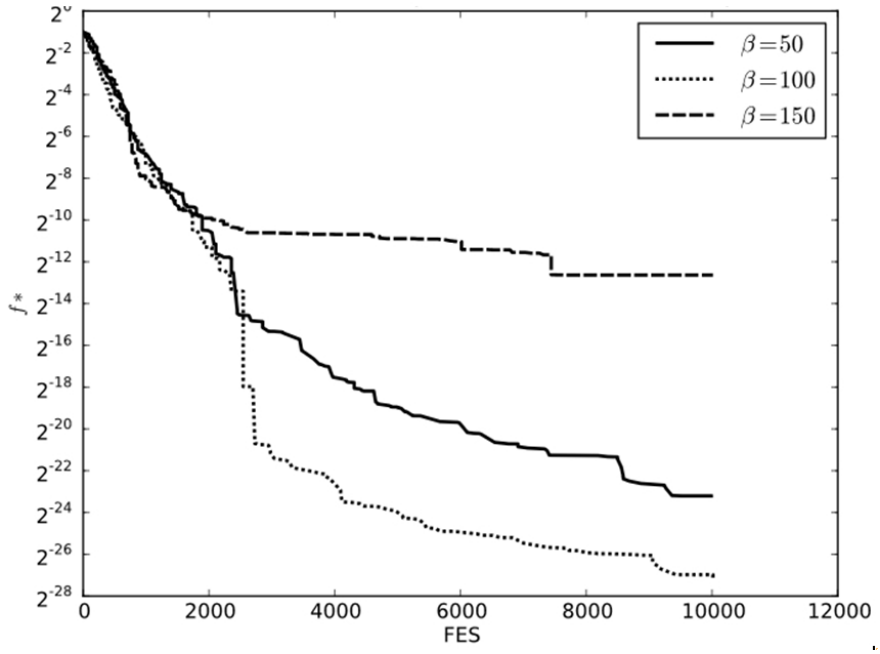


Figure 4.2: The effects of the increment rate divisor (β) on the convergence characteristics of the algorithm tested on 2-D Schaffer Function.

smaller. This is better for the exploiter bat; however the explorer bat will need more time to find the optimal solution.

4.2 The behavior of the virtual bats

Both bats start searching from a random position with random direction, default wavelength, and frequency. At the beginning, they both start the search like an explorer bat. According to the best positions found in their first attempts, one of them will become explorer bat and the other exploiter bat. To assign their roles, we find the best solutions h_* of the bats from their search scopes H and compare them. After determining their roles, there are three possible actions that a bat may take by comparing x_i^t with h_* for its next step $(t+1)$:

1. If the best position of search scope h_* is better than the bat's current position x_i^t, the bat will fly to this position (4.9). And also its direction will be changed towards the next

position (4.10) (see line 10-12 in Algorithm 2). The bat will become the exploiter bat, its wavelength will be shortened (4.7), and its frequency will be increased (4.6). As long as it has a better position in its sound waves scope it will increase intensification of the search. To avoid very small or big $\mathbf{V}_i^{t+1}$ we normalized it by using Eg.4.11. Thus, when $\mathbf{V}_i^{t+1}$ approaches zero, it will be always between $[0, 1]$.

$$x_i^{t+1} = h_* \tag{4.9}$$

$$\mathbf{V}_i^{t+1} = |\mathbf{x}_i^{t+1} - \mathbf{x}_i^t| \tag{4.10}$$

$$\hat{\mathbf{v}}_i = \frac{\mathbf{V}_i^{t+1}}{\|\mathbf{V}_i^{t+1}\|} \tag{4.11}$$

If h_* is worse than the bat's current position, the algorithm checks whether the current position x_i^t is the best one ever found x_{gbest} or not.

2. If the current solution is not the best solution ever found then the bat becomes the explorer bat, changes its direction randomly, increases the wavelength (4.7), and decreases the frequency (4.6) expanding the search scope (see line 13-15 in Algorithm 2). These actions help the bat keep exploring the search space without getting trapped in a local optima and provides a random walk.

3. If the bat is already on the best found position (x_{gbest}), the bat will become the exploiter bat. The wavelength will be minimized $(\lambda_i^{t+1}) = (\lambda_{min})$, the frequency will be maximized$(f_i^{t+1}) = (f_{max})$, and the search direction will be changed randomly (see line 16-18 in Algorithm 2). As a result, the bat can increase the intensification of the search around the best position.

Based on the virtual bat's behavior, the basic steps of the algorithm for minimizing $f(x)$ are shown in Algorithm 2,

Algorithm 2 DVBA pseudo code

```
Objective function f(x), x = (x_1, ..., x_d)^T
Initialize the bat population x_i (i = 1, 2) and v_i
Initialize wavelength λ_i and frequency f_i
Initialize the number of the waves
while (t < Max number of iterations) do
    for each bat do
        Create a sound waves scope
        Evaluate the solutions on the waves
        Choose the best solution on the waves, h_*
    if (f(h_*) < f(x_i)) then
        Move to new solution
        Decrease λ_i and increase f_i
    else if (f(x_i) > f_gbest) then
        Change the direction randomly
        Increase λ_i and decrease f_i
    else if (f(x_i) = f_gbest) then
        Minimize λ_i and maximize f_i
        Change the direction randomly
    end if
    Rank the bats and find the current best x_gbest
end while
```

where f_{gbest} is the global best solution and d is the number of dimensions.

4.3 The effects of the major parameters on DVBA

For optimizing a problem, system designer faces two important difficulties. First one is finding the best algorithm for the problem and the other one is setting the parameters of the algorithm. Generally algorithms are expected to perform the same success for any type of optimization problem and that shows the success of the algorithm. However, that is a difficult task for the algorithms, since there are too many types of optimization problems. Because of that, most of the algorithms have parameters to be tuned for efficiency. According to the type of the optimization problems these parameters can be set differently.

There are very few algorithms which are not depending on the parameters. Most of the algorithms depend on one or more parameters which have to be set to some values [70].

The choice of parameters can have significant impact on the effectiveness of the optimization algorithm. Specially, if the optimization problem is discontinuous, high-dimensional, noisy, or multimodal, setting the parameters is one crucial factor for efficiency [16].

In DVBA, there are two main parameters which affect its performance [96]. They are step size divisor and search scope size parameters (number of search points). While step size divisor changes the bats speed, search scope size parameters makes the search scope of the bats wider or narrower.

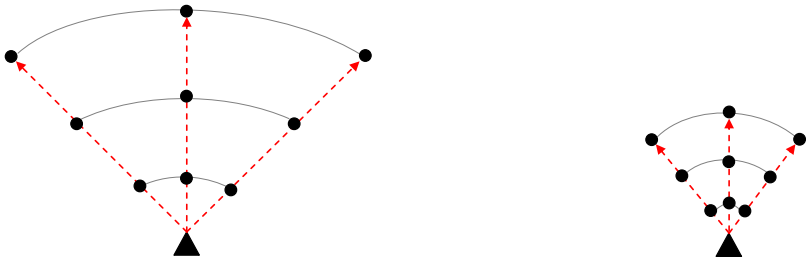
In this section, we empirically study the effects of the major parameters on DVBA and give a list of good choices of parameters for various optimization scenarios. Five different well-known benchmark functions are selected as testing functions.

4.3.1 Analyzing the number of search points and step size divisor effects on the performance of DVBA

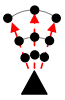
In DVBA, the virtual bat's search scope size is proportional with its emitted sound wavelength (λ) and frequency (f). In DVBA, increasing the wavelength will increase the distance between the search points and reducing the frequency which is inversely proportional with the wavelength, will increase the angle between the search vectors (A_j) (4.1) (See Fig.4.1c). That will help the explorer bat to widen its search scope as shown in Fig.4.3a. Controversially, reducing the wavelength will increase the frequency and that will create narrow search scope which is used for the exploiter bat as in Fig.4.3c. The changes of the frequencyf_i and the wavelength $_i$ at the time step $t+1$ are given by

$$f_i^{t+1} = f_i^t \pm \rho \tag{4.12}$$

$$\lambda_i^{t+1} = \lambda_i^t \pm \rho \tag{4.13}$$



(a) $\beta = 25$ (b) $\beta = 100$



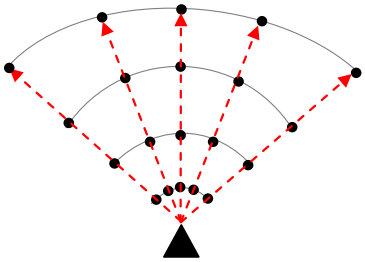
(c) $\beta = 300$

Figure 4.3: The effect of step size divisor on the width of bat's search scope sizes in DVBA

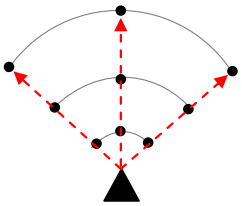
$$\rho = mean(\frac{\mathbf{U} - \mathbf{L}}{\beta}), \qquad \{\beta \in \Re : \beta > 0\} \tag{4.14}$$

where ρ and β are positive real constants, U is the upper bounds and L is the lower bounds of the search space, and ρ is used as increment rate for frequency and wavelength. β is used as increment rate divisor. It is clear that, wavelength and frequency changes related directly with β. Very high β will cause very small distance between the search points and narrow search scope. That is good for the exploiter bat to make intensive exploitation. However, the explorer bat may trap in local minima because of narrow search scope. On the other hand, very small β will increase the distance between the search points and the bats will have large search scope. This will help explorer bat to explore better the search space, but the exploiter bat will not able to have an intensive exploitation search. So, choosing the right value of β is very important for efficiency.

In addition, the number of search points in the search scope has a major effect on the performance of the algorithm as well. The changes of the number of search points will affect



(a) Virtual bat with 20 search points



(b) Virtual bat with 9 search points

(c) Virtual bat with 2 search points

Figure 4.4: The different search scope sizes of a virtual bat in DVBA

the bat's search scope size directly as shown in Fig.4.4. In most tests, it has been decided as 30 after some experiments. However this might not be the best choice for different scenarios. In this section, the effect of the number of search points is investigated for different types of optimization problems.

To study the effects of the number of the search points and the step size divisor some experiments are designed in this section. A set of bound constrained benchmark functions is used to investigate the effect of the two parameters. Rastrigin and Griewangs are multimodal functions. Quartic is a noisy function. Sphere and Rosenbrock are unimodal functions. Functions were tested with 2 and 30 dimensions to get more accurate results. Functions details are given in Table.4.1.

In order to investigate whether the DVBA scales well or not, different step size divisors and search points are used for each function with different dimensions. They are step size divisors (β) of 20, 50, 100, 200, and 500 Table.4.2. The number of search points varied from

Table 4.1: Description of the benchmark functions used. Here D: Dimensionality of the functions, C: function characteristics with values – U: unimodal, M: Multimodal, and N: Noisy.

No	Name	Formula	C	f_{min}	Search Space
f_1	Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	U	0	$(-100, 100)^D$
f_2	Rosenbrock	$f_2(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	U	0	$(-100, 100)^D$
f_3	Quartic	$f_3(x) = \sum_{i=1}^{d} ix_i + rand[0, 1)$	N	0	$(-1.28, 1.28)^D$
f_4	Rastrigin	$f_4(x) = 10D + \sum_{i=1}^{D} [x_i^2 - 10cos(2\pi x_i)]$	M	0	$(-5.12, 5.12)^D$
f_5	Griewangk	$f_5(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$	M	0	$(-600, 600)^D$

4 to 150 points Table.4.3. In Table.4.2 and Table.4.3, the results are shown in terms of mean and standard deviation (SD). The results are found over 30 independent runs. The final best function values are used to calculate the mean and the SD of different scenarios of DVBA. Maximum number of function evaluations (FES) is set to 10.000 when solving 2-D functions, 100.000 for 30-D problems. For the tests in Table.4.2, the number of search points is set to 30 and for the tests in Table.4.3, the step size divisor (β) is set to 100.

Fig.4.5 illustrates that the average fitness varied with β from 20 to 500 and search points from 4 to 150 for 4 functions. As shown in Fig.4.5, too small or too large β will generally lead to bad results. This is due to the fact that, a smaller β means that the distance between search points becomes larger (Fig.4.3a), bats might easily pass over but without catching good solutions. On the contrary, a larger β will reduce the distance between the search points that facilitates local exploitation and bats may not have sufficient opportunity to explore more space local regions. Therefore, the bats could be easily trapped in local optima.

It is clear from Fig.4.5 that increasing the number of search points will lead to bad results as well. That is because the algorithm expends too many FEs and may not have sufficient time to find the optimal solution, unable to move far enough to reach better positions globally.

By examining Sphere and Rosenbrock function of high dimension in Table.4.2, it can be said that DVBA performs poorly when β is chosen very small. For the 2-D unimodal functions, using different β does not make significant difference on the DVBA's performance. For the Quartic

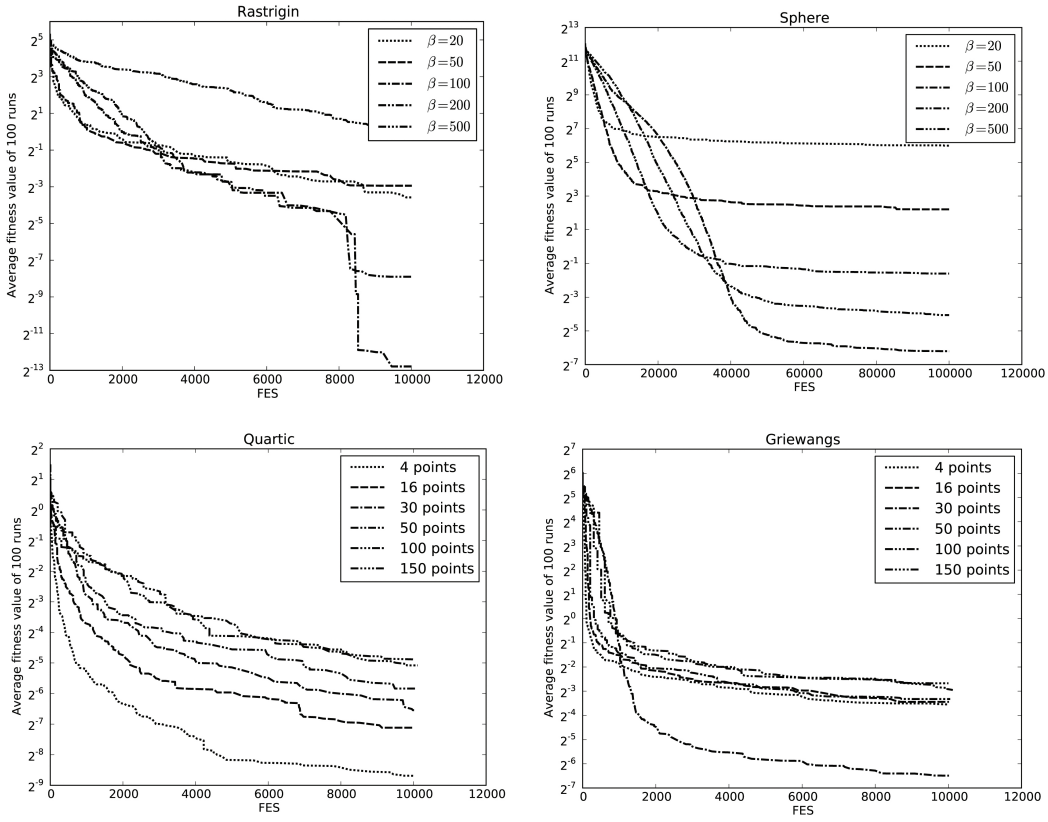


Figure 4.5: Convergence Characteristics of DVBA for different parameters. Sphere and Quartic are in 30 dimensions. Rastrigin and Griewangs are in 2 dimensions.

function, the fitness values are very close and no rules can be extracted. For Griewangs and Rastrigin function of high dimension, scaling β after 100 doesn't affect the DVBA's performance significantly. However, when these functions are in 2-D form, we can say that the best choice of β is around 100.

As reported in Table.4.3, the DVBA lacks of global search ability when the number of search points are increased. From Table.4.3, it can be concluded that for both high and low-dimensions of Sphere, Rastrigin, and Rosenbrock functions, the best value of search points is 30. For Rosenbrock of low dimension and Quartic of high dimension, it can be said that the best value

of search points is 4.

In summary, we can say that taking larger value of β and search points smaller than 30 will generally give good results for the most problems. As a future work, before the algorithm runs, the parameters can be optimized according to the problem.

Table 4.2: Influence of step size divisor β. D denotes the dimensions.

DVBA		Sphere		Rosenbrock		Quartic		Rastrigin		Griewangs	
with	D	2	30	2	30	2	30	2	30	2	30
$\beta = 20$	Mean	8.1E-03	63.4	9.5E-04	68.43	1.3E-02	0.25	8.3E-02	282.4	2.79	834.9
	SD	5.6E-03	5.0	1.5E-03	11.31	9.7E-03	7.3E-02	6.8E-02	24.1	1.54	95.1
$\beta = 50$	Mean	1.1E-03	4.6	1.2E-04	57.38	1.0E-02	0.22	0.13	262.9	0.29	205.9
	SD	1.0E-03	1.0	1.6E-04	28.61	7.9E-03	7.3E-02	0.30	26.7	0.18	23.6
$\beta = 100$	Mean	1.7E-06	1.4E-02	4.4E-04	31.19	1.4E-02	0.73	1.4E-04	186.7	9.7E-03	1.15
	SD	2.2E-06	2.3E-03	9.7E-04	10.84	1.1E-02	0.19	1.8E-04	40.9	6.9E-03	1.9E-02
$\beta = 200$	Mean	1.2E-04	3.3	7.1E-04	28.08	1.5E-02	0.19	4.1E-03	199.0	2.7E-02	1.89
	SD	1.6E-04	5.6E-02	1.2E-03	1.55	1.2E-02	6.4E-02	5.1E-03	43.2	1.4E-02	8.6E-02
$\beta = 500$	Mean	4.8E-05	5.9E-02	1.9E-02	26.70	3.6E-02	0.29	0.78	254.0	6.8E-03	1.03
	SD	5.5E-05	9.1E-03	2.1E-02	2.22	3.3E-02	4.7E-02	1.18	47.4	4.4E-03	3.4E-03

Table 4.3: Influence of number of search points

DVBA		Sphere		Rosenbrock		Quartic		Rastrigin		Griewangs	
with	D	2	30	2	30	2	30	2	30	2	30
4 points	Mean	8.2E-05	0.90	8.9E-06	27.78	2.4E-03	3.8E-02	1.6E-03	211.8	9.1E-02	14.11
	SD	6.7E-05	0.11	1.4E-05	2.25	1.9E-03	6.6E-03	1.7E-03	30.4	4.2E-02	0.87
16 points	Mean	1.9E-04	1.02	8.5E-05	35.38	7.2E-03	0.14	3.7E-03	144.9	7.4E-02	15.35
	SD	1.6E-04	0.15	1.5E-04	18.10	4.4E-03	3.2E-02	7.8E-03	19.6	4.9E-02	1.32
30 points	Mean	2.2E-06	1.2E-02	4.6E-04	28.11	1.1E-02	0.69	7.4E-05	117.5	1.1E-02	1.14
	SD	2.1E-06	1.4E-03	8.4E-04	0.91	6.6E-03	0.16	8.6E-05	10.7	4.3E-03	1.9E-02
50 points	Mean	2.8E-04	1.28	1.2E-03	42.01	1.7E-02	0.32	6.4E-02	132.5	9.1E-02	15.20
	SD	2.6E-04	0.27	3.6E-03	22.68	1.4E-02	0.11	0.23	21.1	5.6E-02	2.03
100 points	Mean	4.1E-04	1.56	3.6E-03	38.65	3.4E-02	0.51	0.11	134.9	0.10	16.32
	SD	5.6E-04	0.33	8.1E-03	12.92	2.2E-02	0.19	0.22	21.9	4.9E-02	2.88
150 points	Mean	6.6E-04	2.31	5.2E-03	94.19	2.3E-02	0.93	0.51	132.8	0.16	16.95
	SD	1.0E-03	0.62	1.4E-02	21.02	1.8E-02	0.23	0.65	21.6	8.0E-02	1.86

CHAPTER 5

NUMERICAL EXPERIMENTS AND RESULTS

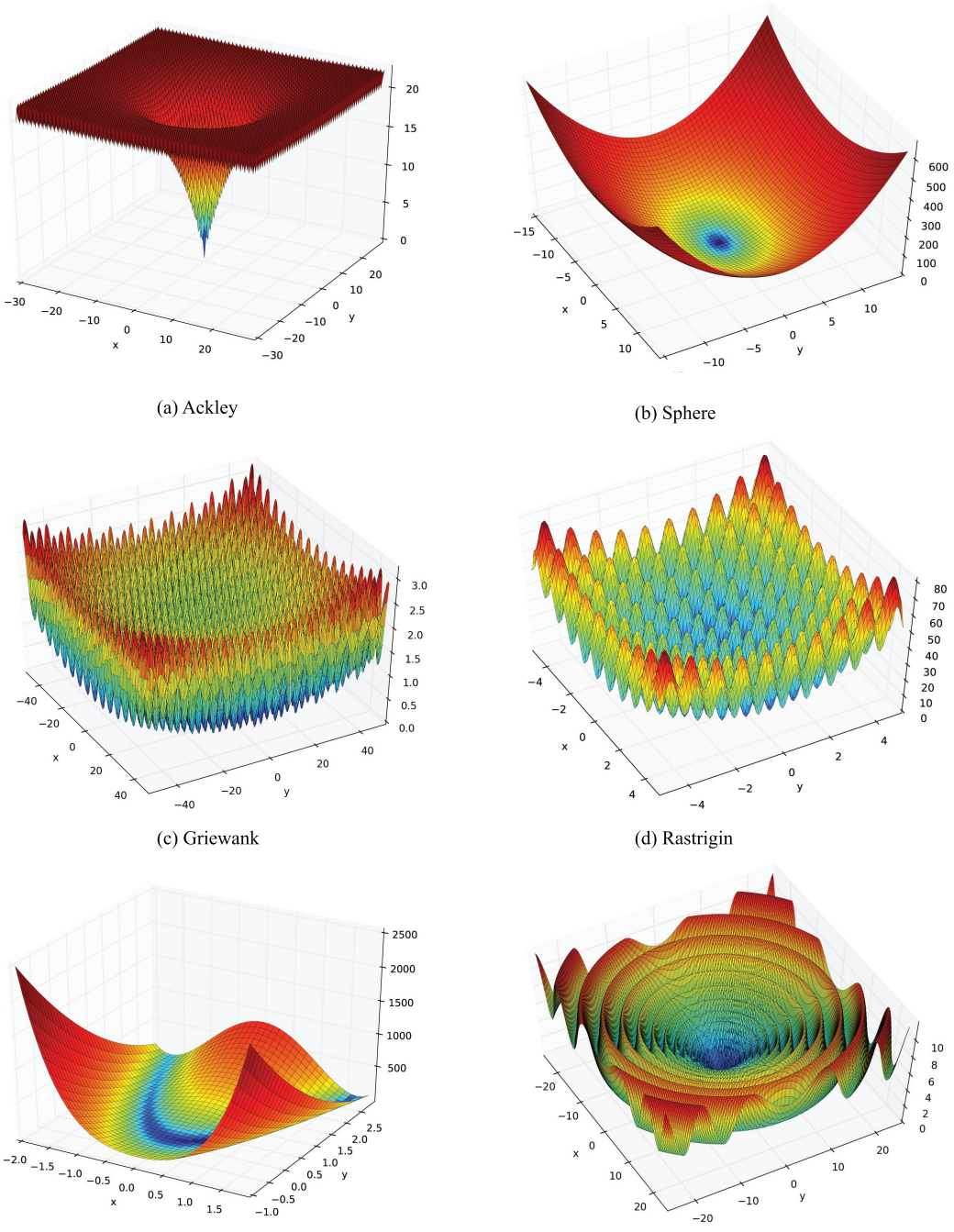
This part of the chapter is devoted to show different aspect of the experimentation carried out to present detailed information about the performances of the algorithms. Firstly, we will define the test functions,then, comparison algorithms and experiments will be presented. Then, we will analyze the experiments results. Finally, the performance of DVBA will be tested on supply chain cost with other well-known algorithms.

5.1 Optimization test functions

To evaluate the performance of the proposed algorithm DVBA, algorithms are run to minimize a set of 6 scalable classic benchmark functions and 30 CEC 2014 test functions as shown in Table 5.1.

Benchmark problems in CEC 2014 test suit are developed by Liang to the approaches, algorithms and techniques for solving real parameter single objective optimization without making use of the exact equations of the test functions. In CEC 2014, benchmark problems have several novel features such as novel basic problems, composing test problems by extracting features dimension-wise from several problems, graded level of linkages, rotated trap problems, and so on [56].

As it is explained in [58], most of the optimization test functions have global optima at the center of the search space and local optima lie along the coordinate axes. These



(a) Ackley

(b) Sphere

(c) Griewank

(d) Rastrigin

(e) Rosenbrock

(f) Schaffer

Figure 5.1: Classic optimization test functions [26]

Table 5.1: Descriptions of the test functions

Groups	**No.**	**Functions**	**Search range**	**Global optimum**	**D**
Classic Test Functions	F1	Rastrigin	[-5.12, 5.12]	0	2
	F2	Rosenbrock	[-30,30]	0	2
	F3	Levy N.13	[-10,10]	0	2
	F4	Griewank	[-600, 600]	0	2
	F5	Schaffer Function 2	[-100,100]	0	2
	F6	Zakharov	[-5, 10]	0	2
CEC 2014 Test Functions					
Unimodal Functions	*f1*	Rotated high conditioned elliptic	[-100, 100]	100	30
	f2	Rotated bent cigar	[-100, 100]	200	30
	f3	Rotated discus	[-100, 100]	300	30
Simple Multimodal Functions	*f4*	Shifted and rotated rosenbrock's	[-100, 100]	400	30
	f5	Shifted and rotated ackley's	[-100, 100]	500	30
	f6	Shifted and rotated weierstrass	[-100, 100]	600	30
	f7	Shifted and rotated griewank's	[-100, 100]	700	30
	f8	Shifted rastrigin's	[-100, 100]	800	30
	f9	Shifted and rotated rastrigin's	[-100, 100]	900	30
	f10	Shifted schwefel's	[-100, 100]	1000	30
	f11	Shifted and rotated schwefel's	[-100, 100]	1100	30
	f12	Shifted and rotated katsuura	[-100, 100]	1200	30
	f13	Shifted and rotated happyCat	[-100, 100]	1300	30
	f14	Shifted and rotated HGBat	[-100, 100]	1400	30
	f15	Shifted and rotated expanded griewank's plus Rosenbrock's	[-100, 100]	1500	30
	f16	Shifted and rotated expanded scaffer's F6	[-100, 100]	1600	30
Hybrid Function 1	*f17*	Hybrid function 1 (N=3)	[-100, 100]	1700	30
	f18	Hybrid function 2 (N=3)	[-100, 100]	1800	30
	f19	Hybrid function 3 (N=4)	[-100, 100]	1900	30
	f20	Hybrid function 4 (N=4)	[-100, 100]	2000	30
	f21	Hybrid function 5 (N=5)	[-100, 100]	2100	30
	f22	Hybrid function 6 (N=5)	[-100, 100]	2200	30
Composition Functions	*f23*	Composition function 1 (N=5)	[-100, 100]	2300	30
	f24	Composition function 2 (N=3)	[-100, 100]	2400	30
	f25	Composition function 3 (N=3)	[-100, 100]	2500	30
	f26	Composition function 4 (N=5)	[-100, 100]	2600	30
	f27	Composition function 5 (N=5)	[-100, 100]	2700	30
	f28	Composition function 6 (N=5)	[-100, 100]	2800	30
	f29	Composition function 7 (N=3)	[-100, 100]	2900	30
	f30	Composition function 8 (N=3)	[-100, 100]	3000	30

problems have been solved in the CEC test suite by shifting global optima and rotating the test functions. Benchmark functions reflected real world problems and became more challenging for optimization problems in CEC test suite. CEC 2014 test functions include 3 rotated unimodal, 13 shifted and rotated multi-modal, 6 hybrid, and 8 composition test functions All the functions are minimization problems. The details about the functions are shown in Table 5.1. In Figure 5.1, some of the functions are shown.

5.2 Experimental Setting

All experiments were executed on a standard PC with Windows 8, Intel(R) Core(TM) i5-3470 CPU, 3.20GHz, 8 GB RAM. All the algorithms which were used for comparisons were developed in the Python environment according to the descriptions in their original papers. The results for PSO and BA are compared with the results in [100] and some of the comparison results are given in Appendix A.1. The codes are available in Oguz Altun's Bitbucket repository (`https://bitbucket.org/oaltun/opn`

5.3 Analyzing the performance of DVBA on optimization test functions

This section presents an extensive comparison among the performances of 14 algorithms on two groups of test functions. In the first group, there are 6 classic optimization test functions with the dimensions of 2. In the second group, there are 30 numerical functions from the 2014 Congress on Evolutionary Computation (CEC 2014) Special Session.

5.3.1 Comparison Algorithms

In order to demonstrate the effectiveness of DVBA, it is compared with 13 algorithms in three groups. In the first group, four well known standard optimization algorithms are used. Since the

proposed algorithm DVBA is inspired from bats, DVBA is compared with four state-of-the-art modified BA algorithms in the second group. In the third group, five algorithms are used from the special session at CEC 2014 for comparison. The algorithms in comparison are listed as follows:

Group 1:

- Particle Swarm Optimization (PSO) [47]
- Genetic Algorithms (GA) [31]
- Simulated Annealing (SA) [1]
- Bat Algorithm (BA) [106]

Group 2:

- Local Memory Search Bat Algorithm (LMSBA) [113]
- Novel Adaptive Bat Algorithm (NABA) [42]
- Adaptive Bat Algorithm (ABA) [100]
- Chaotic Local Search-based Bat Algorithm (CLSBA) [59]

Group 3:

- FWA-DM [112]
- L-SHADE [91]
- NRGA [110]
- OPTBees [21]
- b6e6rl [74]

Here, GA and PSO are chosen because they are used widely in many optimization problems with great success. Since the proposed algorithm DVBA is inspired from bats, BA is chosen as well, and explained in section 3 with details to show differences with DVBA. SA is not a population based algorithm like other compared algorithms. Most of the population based algorithms suffer from the problem of premature convergence. We wanted to see how the performance of DVBA compares to single-point algorithms like SA.

There are many versions of the PSO, GA, and SA; however, we prefer to use the standard version of the algorithms for comparisons. For PSO we set the inertia weight $w = 1$ and the

acceleration coefficients $c_1 = c_2 = 2$ [47]. For genetic algorithms, no elitism has been used with the mutation probability of $p_m = 0.05$ and crossover probability of 0.95 [31]. As for the BA, parameters are set as follows: $\alpha = \gamma = 0.5$, frequency is in the range $[0, 2]$, and $A_0 = 0.5$ [106]. For SA, we set the initial temperature at 100 and cooling factor to 0.2 [1]. SA re-starts the search from a random position after a local search fails in this version.

As in most of the stochastic algorithms, the standard BA suffers from the premature convergence problem, and it needs improvements in exploration. The random walk size and the pulse rate parameters play very important roles in the exploration ability of BA. As a result, most of the researchers have focused on these parameters. Recently, Local Memory Search Bat Algorithm (LMSBA) [113], Novel Adaptive Bat Algorithm (NABA) [42], Adaptive Bat Algorithm (ABA) [100], and Chaotic Local Search-based Bat Algorithm (CLSBA) [59] look toward improving BA's exploration ability. These algorithms have been tested on classic optimization test functions in their original paper, they have not been compared yet on a difficult test suit, like CEC 2014. Hence, four BA variants and DVBA are compared on CEC 2014 test suit in the second group.

In group 3, FWA-DM, L-SHADE, and b6e6rl are the-state-of-the-art Differential Evolution (DE) variants which are lately tested on CEC 2014 test suits successfully. NRGA is an improved versions of Genetic Algorithm and OPTBees is an algorithm which is inspired by the collective decision-making of bee colonies.

- FWA-DM (Fireworks Algorithm with Differential Mutation): FWA was introduced by Tan and Zhu in 2010 [89] as a new metaheuristic method that was based on the fireworks explosion in the sky at night. When a firework explodes, a shower of sparks appears around the firework. In this way, adjacent area of the firework is searched. By controlling the amplitude of the explosion and the number of explosion sparks, the ability of local search for FWA is guaranteed.

In 2014, Chao Yu introduced Differential Evolution (DE) mutation operator to fireworks algorithm (FWA) and a new algorithm was formed, namely Fireworks Algorithm with Differential Mutation (FWA-DM) [112]. In FWA-DM, firstly the individuals are initialized

randomly and marked as POP1. Secondly, a spark is produced around each individual within a certain amplitude. These sparks form a population POP2. Thirdly, the individuals are compared in POP1 and POP2 correspondingly and the ones with better fitness values are kept and used to form a new population marked as POP3. In this step, Chao applied the mutation and crossover operators in DE algorithm to POP3 and a new population is generated as POP4. Finally, POP1 is regenerated by applying the selection operator to POP4.

- L-SHADE: Success-History based Adaptive Differential Evolution (SHADE) is an improved versions of JADE [114] which is an adaptive Differential Evolution algorithm. In 2014, Ryoji and Alex intruduced L-SHADE [91] which further extends SHADE with Linear Population Size Reduction (LPSR). LPSR is a simple deterministic population resizing method which continually reduces the number of population according to a linear function.

It is well-known that the search performance of the algorithms depends on control parameters. DE has three main control parameters, which are the number of population NP, scaling factor F, and crossover rate CR. To apply DE to a real-world problem, it is needed to tune the parameters in order to have maximum performance. However, it is a significant problem in practice. To overcome this problem, success-history based adaptation mechanism have been studied by researchers [90]. This mechanism uses a historical memory which stores parameters values that have performed well in the past. Then it generates new values of the parameters by directly sampling the parameter space close to one of these stored parameters.

In SHADE, the CR and F are automatically adjusted by success-history adaptation, but the number of population NP remains constant throughout the search. To further enhance the performance of SHADE, Ryoji and Alex proposed L-SHADE which includes an adaptive population resizing method to SHADE. So, the number of population size reduces continually by using LPSR method. By reducing the population size throughout the search they assumed that the rate of convergence will be faster.

- NRGA (Non-Uniform Real-coded Genetic Algorithm): Different techniques have been proposed in literature for performing selection step in GA. Non-Uniform Real-codded Genetic Algorithm (NRGA) [110] is one of the latest proposed improved GA that uses

tournament selection operator for doing selection. The tournament is performed between certain number of randomly selected individual and the better one is chosen for crossover. For the crossover step, SBX crossover operator is used which is proposed in [15].

NRGA, not only modified the classic selection and crossover steps but also added a non-uniform mapping operator for making further modification in the population obtained after mutation. This operator pushes the population towards the better solution obtained so far.

- OPTBees: OPTBees [21] is a swarm based algorithm which was proposed in 2014 for optimization continuous spaces. It is inspired by the collective decision-making of bee colonies. The difference of OPTBees from other swarm based algorithms is the use of different types of agents with different roles. According to the problem these roles might change for each agent. In OPTBees, three different types of agents are used: 1- $recruiters$, that recruit bees for exploiting a promising region; 2- $scouts$, that randomly search the search space; 3- $recruited$, that are recruited by recruiters to exploit the chosen region. These types of bees are represents the active bees. There are other bees in the hive with no specific task assigned to them which are called inactive bees.

The active bees search for high quality food source. According to the quality of the food source they discovered, bees are classified as recruiters or non-recruiters. The recruiter bees recruit some of the non-recruiter bees to increase the exploitation around the high quality food source found so far and non-recruiter bees keep flying around the find food sources. The scout bees randomly fly around for a better food sources. If the active bees find a large high quality food sources, some of the inactive bees become active and help other bees to explore faster this region.

- b6e6rl: Radka [74] was proposed a new variant of competitive differential evolution (CDE) with the controlled restart in 2014. By using controlled restart, he wanted to avoid from the stagnation of the search. In DE, it was observed that a minimum fitness remained the same for thousands of generations until the search stops. The stagnation can be caused by small diversity of population or trapping in a local minimum. In this situation, Radka proposed

to give population a new impulse.

To detect the stagnation, the difference of maximum and minimum values of the objective function and the estimated maximum distance among the points in the current population are used. A very small difference can indicate a trapping at a local minimum.

Radka applied the controlled restart technique in one of the competitive-adaptation variant of DE(CDE) denoted b6e6rl which was found as one of the most efficient among the other variants. The b6e6rl uses two DE strategies[1] which are DE/randrl/1/bin and DE/randrl/1/exp.

For the algorithms in the second and third groups, parameter settings of the algorithms are the same as in their original papers.

5.3.2 Comparison Experiments

This work aims to test the quality of the final solution and the convergence speed at the end of a fixed number of function evaluations (FEs). The comparisons are done under two section. In the first section, PSO, BA, GA, and SA are compared with DVBA on 6 classic optimization test functions and CEC 2014 test functions. In the second section, DVBA is compared with four state-of-the-art modified BA versions and five algorithms from special session at CEC 2014 on all 30 CEC 2014 test functions.

5.3.3 Comparison the algorithms in Group 1

The algorithms from the first group are tested on classic optimization test functions and CEC 2014 test functions; the results are shown in Table 5.2, 5.3, 5.4, 5.5, 5.6, and 5.7. Each test function was considered $2-D$ in classic optimization test functions. The dimensions are set to 30 for CEC 2014 test functions. Maximum number of function evaluation (FEs) is set to 1×10^4 for $2-D$ problems and 3×10^5 for $30-D$ problems. Algorithms were tested with 30 independent runs for each test functions in order to compile comprehensive data. After these runs the best fitness values (BFV), the worst fitness values (WFV), the mean of the results (Mean) and

[1] `http://www.icsi.berkeley.edu/~storn/code.html`

the standard deviation (STDEV) are recorded for each algorithms and test functions. Then the success rate (SR) is calculated as the number of successful trials (st) divided by the total number of trials (tt) (5.1). In this work, a trial is called successful when its closeness to the predefined optimal position is less then 1.0E-05. The best results are typed in bold.

$$SR = \frac{st}{tt} x100\% \tag{5.1}$$

SR has been calculated only for the first group of the test functions in Table 5.2. For the other groups, functions were tested in 30-100 dimensions and the SRs were not close enough to the optimal positions hence they are not shown in other tables.

The comparison of accuracy of algorithms for each test function group and the overall success rates have been shown in Table 5.8 and Table 5.9. The overall success (OSM) ratios for Mean are calculated for each algorithm as the number of functions which algorithm got the best Mean *(nfbm-number of functions with best Mean)* divided by the number of total functions (tf) (5.2). Same formula applied to get overall success ratios for BFV, WFV, and STDEV in Table 5.9.

$$OSM = \frac{nfbm}{tf} x100\% \tag{5.2}$$

Fig. 5.2, 5.3, 5.4, and 5.5 illustrate the convergence characteristics in terms of the best fitness value of the median run of each algorithm, which is plotted using a logarithmic scale in order to reduce the biggest and smallest values in the whole optimization process.

Empirical results of the algorithms on classic optimization test functions

As Table 5.2 shows, the test functions in group 1 are mixture of multimodal, unimodal, separable, and non-separable functions. It can be seen from Table 5.2 that, most algorithms can locate the global optima with high success rate for 2-D functions. Since the SA has been using single particle with many starts, its performance was very poor in terms of SR among compared algorithms in group 1. It can be observed from Table 5.2 that DVBA found the global optima with higher success ratio than other algorithms for these functions, followed by BA then GA.

Table 5.2: Optimization results for the classic test functions.

	Function	**Algo.**	**BFV**	**WFV**	**Mean**	**STDEV**	**SR**
F1	Rastrigin	PSO	4.26E-5	1.226	0.18	**0.311**	73.3%
		SA	9.4E-3	1.243	0.4415	0.36322	0.0%
		GA	5.912E-5	1.989	0.7022	0.5675	63.3%
		BA	4.75E-4	1.0055	0.5325	0.4963	63.3%
		DVBA	**3.141E-6**	**0.9956**	**0.1331**	0.3381	**90.0**%
F2	Rosenbrock	PSO	1.625E-6	3.015E-4	1.141E-4	9.18E-5	63.33%
		SA	5.880E-5	1.072E-2	2.435E-3	2.972E-3	6.66%
		GA	2.648E-6	2.713E-2	5.369E-3	7.528E-3	23.3%
		BA	5.644E-7	4.781E-5	1.679E-5	**1.246E-5**	**100.0**%
		DVBA	**1.451E-8**	**9.608E-6**	**1.264E-5**	2.344E-5	**100.0**%
F3	Levy N.13	PSO	5.287E-5	4.645E-2	6.91E-3	1.178E-2	6.66 %
		SA	2.637E-3	1.77E-1	5.985E-2	5.113E-2	0.0 %
		GA	1.391E-6	**6.168E-5**	**2.078E-5**	**1.953E-5**	**100.0** %
		BA	1.085E-5	3.830E-4	1.258E-4	1.001E-4	63.3 %
		DVBA	**7.111E-7**	8.726E-5	3.158E-5	4.913E-5	**100.0** %
F4	Griewank	PSO	1.232E-3	1.290E-1	5.561E-2	3.390E-2	0.0%
		SA	4.254E-2	4.543E-1	1.583E-1	9.251E-2	0.0%
		GA	**9.262E-5**	**1.175E-2**	5**.798E-3**	**3.509E-3**	**3.3** %
		BA	2.471E-4	2.057E-2	8.791E-3	5.559E-3	0.0%
		DVBA	4.505E-4	6.418E-2	2.362E-2	1.429E-2	0.0 %
F5	Schaffer Function 2	PSO	2.206E-8	1.691E-2	2.526E-3	3.686E-3	56.6 %
		SA	1.416E-4	3.662E-2	1.674E-2	1.187E-2	6.6 %
		GA	2.623E-6	3.057E-1	8.923E-2	9.241E-2	20.0 %
		BA	3.520E-7	6.230E-5	1.235E-5	1.231E-5	**100.0** %
		DVBA	6**.818E-9**	**1.330E-6**	**2.552E-7**	**3.026E-7**	**100.0** %
F6	Zakharov	PSO	1.079E-4	1.313E-2	2.163E-3	2.721E-3	0.0 %
		SA	6.419E-4	9.751E-2	3.004E-2	2.499E-2	0.0 %
		GA	8**.470E-9**	1.457E-2	4.859E-4	2.615E-3	86.6 %
		BA	9.464E-7	**1.828E-4**	**4.431E-5**	**4.292E-5**	**96.6** %
		DVBA	5.495E-7	4.480E-4	7.049E-5	9.559E-5	86.6 %

The functions F1 and F4 are separable, highly multi-modal problems, where the number of local minima exponentially increases as the dimension increases however locations of the minima are regularly distributed. For the function F1 which is a hard test problem, DVBA has the best value in terms of all properties except STDEV. However for the function F4, none of the algorithms performed successfully and there were not any significant difference between their performance. F4 has many widespread local minima and algorithms can be trapped easily in one of these local minima. The functions F2 and F6 are unimodal plate and valley-shaped functions. F2 is a popular test function for gradient-based optimization algorithms. Although F2 is unimodal, convergence to the minimum is difficult [73]. Both BA and DVBA found the global optima with 100% successful ratio for F2 and F5, but DVBA has the best BFV, WFV and the Mean. For the function F6, BA performed better than other algorithms, but the difference between BA and DVBA is negligible in terms of the Mean.

Fig.5.2 shows the convergence map of the five algorithms for the functions from group 1. It is observed that PSO, GA, and BA converge faster than SA and DVBA, but after a learning period, the convergence speed of DVBA abruptly accelerates. If the number of function evaluations (FEs) is increased slightly, it may be argued that DVBA can outperform other population based algorithms. Since DVBA starts the search by only two bats, it is normal for DVBA to stay behind from PSO, BA, and GA in terms of convergence speed, because they start the same search by using 30 particles.

Empirical results on unimodal functions from CEC 2014

The functions f_1, f_2, and f_3 are unimodal and non-separable plate shape problems. From Table 5.3, it can be seen clearly that these functions are very hard to optimize. Although they have just one minimum, same as F2 (rosenbrock), convergence to the minimum is difficult. The algorithms did not show a significant success for these functions. DVBA has demonstrated a better ability of global searching for functions f_1, f_2, and f_3, followed by GA. For the function f_2, the performance of DVBA outperforms significantly (almost 50% better) PSO, SA, GA, and BA as the test results show. It shows that the explorer bat in DVBA flied through the global

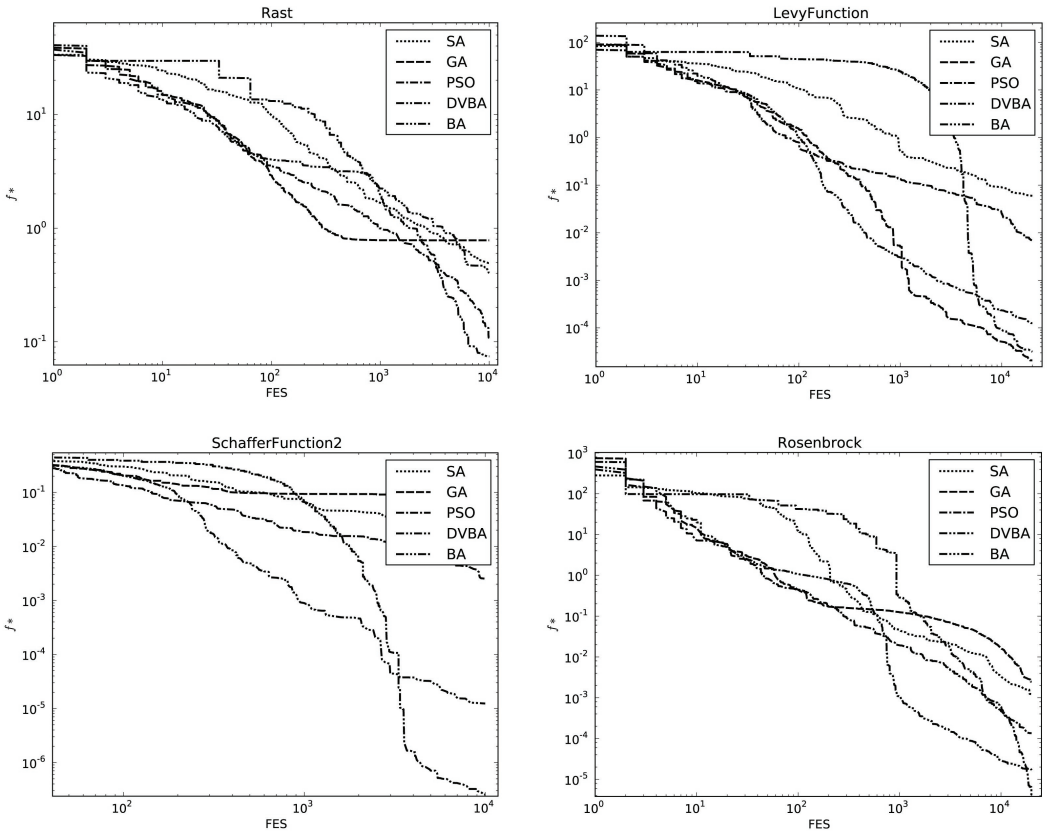


Figure 5.2: Convergence characteristics of SA, GA, PSO, DVBA, and BA on classic test functions.

Table 5.3: Optimization results for unimodal functions in CEC 2014 test suit.

	Function	Algorithms	BFV	WFV	Mean	STDEV
	Rotated	PSO	1.819E+7	1.496E+8	5.999E+7	3.682E+7
	High	SA	5.881E+8	1.193E+9	8.807E+8	1.664E+8
f1	Conditioned	GA	6.091E+5	1.744E+6	1.144E+6	**3.794E+5**
	Elliptic	BA	3.357E+6	1.104E+7	6.944E+6	2.218E+6
	Function	DVBA	**3.457E+5**	**1.644E+6**	**1.047E+5**	4.284E+5
		PSO	1.139E+9	3.153E+9	1.630E+9	5.527E+8
	Rotated	SA	6.488E+10	7.892E+10	7.221E+10	3.955E+9
f2	Bent Cigar	GA	9.873E+6	1.481E+7	1.207E+7	1.398E+6
	Function	BA	1.058E+8	1.546E+8	1.256E+8	1.448E+7
		DVBA	**1.647E+4**	**5.454E+4**	**2.934E+4**	**1.327E+4**
		PSO	3.262E+4	7.567E+4	6.252E+4	1.187E+4
	Rotated	SA	1.128E+5	1.459E+5	1.250E+5	1.006E+4
f3	Discus	GA	7.073E+3	4.916E+4	2.979E+4	1.364E+4
	Function	BA	1.335E+4	2.724E+4	2.212E+4	**4.447E+3**
		DVBA	**6.307E+3**	**2.639E+4**	**1.620E+4**	6.259E+3

best faster than other algorithms do, but the exploiter bat could not finish the exploitation within the maximum FEs. For the 30-D problems $f_1 - f_{30}$, SA has great difficulty in finding the global optima on all problems.

It can be observed from Fig.5.3 that DVBA shows same convergence characteristics again: after a learning period, the convergence speed of DVBA accelerates and provides the best performance. For the function f_1 and f_2, this characteristic of convergence can be seen clearly. For the function f_3 all the algorithms behave same, but DVBA got slightly better results (see Table 5.3).

Empirical results on simple multi-modal functions from CEC 2014

In Table 5.4 and Table 5.5, test results of the algorithms on 13 simple multi-modal functions are presented. For the functions $f_4, f_5, f_7, f_8, f_9, f_{10}$, and f_{11}, DVBA performs much better with the smallest BFV and the Mean. Only for the function f_{10}, PSO shows the best values in terms of BFV, but DVBA performs better in terms of the Mean. GA obtains smaller mean values than PSO, BA, SA, and DVBA for functions $f_4, f_6, f_{12}, f_{13}, f_{14}$, and f_{15}, followed by DVBA. For the functions $f_5, f_8, f_{12}, f_{13}, f_{14}$, and f_{16}, PSO, BA, GA, and DVBA do not show significant difference, they are able to find the global optima, but SA performed poorly.

For F1(Rastrigin), before rotation and shifting, DVBA offers better results than other

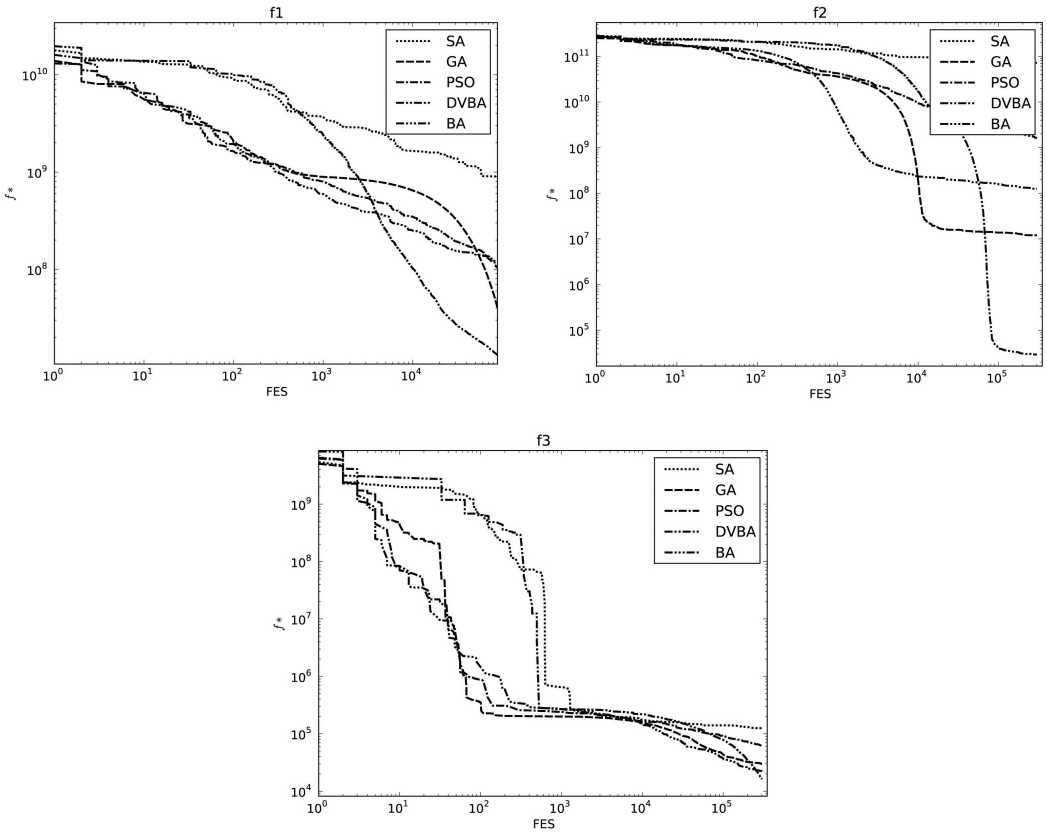


Figure 5.3: Convergence characteristics of SA, GA, PSO, DVBA, and BA on unimodal functions in CEC 2014.

algorithms followed by GA and BA. After rotation and shifting (f_9), DVBA is again able to locate better solution with smaller Mean.

Fig.5.4 shows the convergence process of SA, GA, PSO, DVBA, and BA, conducted on some of the simple multi-modal functions. The simulation results show that, convergence characteristics of algorithms are almost same for functions f_7, f_{13}, f_{14}, and f_{15}. All the algorithms successfully converged before the maximum FEs reached. It is clear that the convergence speed of PSO, BA, and GA better than SA and DVBA. However DVBA converges at an early stage and outperformed the rest four algorithms for function f_5 and f_{11}. BA converges faster than others on f_4 but at the end GA has the smallest average.

The algorithms totally or partially fail in optimizing some functions, but in 53.85% of these multi-modal functions, DVBA performs the best in average, followed by GA with 38.5% (see Table 5.8). This is because that, the dynamic search ability of the bats of DVBA enables its capability of searching global optimum in multi-modal optimization problems. Therefore, we can say that DVBA is a highly competitive algorithm for solving multi-modal optimization problems.

Empirical results on hybrid functions from CEC 2014

In this group of functions, there are six hybrid functions. Hybrid functions are almost same as real-world optimization problems. These functions are more challenging than the functions from previous groups. The concrete expressions and other details of functions are in [55]. Table 5.6 shows optimization results of six hybrid functions with five algorithms.

For hybrid functions, DVBA outperformed the rest four algorithms. Specially, for f_{17}, f_{18}, f_{20}, and f_{21}, DVBA got the smallest values of BFV, WFV, Mean, and STDEV. The algorithms got the best results on f_{19} and f_{22} but they do not outperform significantly each other in terms of the Mean, except SA. In general SA has the worst solutions but got better convergence graph for hybrid functions than it got from unimodal and multi-modal functions.

Fig.5.5 shows the convergence process of SA, GA, PSO, DVBA, and BA, conducted on the functions f_{17}, f_{18}, and f_{20}. Since there was not significant difference on convergence graphics for

Table 5.4: Optimization results for simple multimodal functions-1.

	Function	Algorithms	BFV	WFV	Mean	STDEV
f4	Shifted and Rotated Rosenbrock's Function	PSO	554.0471	911.4058	711.0807	112.6770
		SA	9011.610	17135.149	12290.43	2621.18
		GA	**402.4415**	**405.5765**	**403.7340**	**0.7633**
		BA	480.1962	587.4173	523.1665	36.2369
		DVBA	474.5994	578.8511	507.8066	33.1917
f5	Shifted and Rotated Ackley's Function	PSO	520.5934	520.9324	520.7670	0.0943
		SA	520.8633	521.0279	520.9648	0.0508
		GA	520.8638	521.0195	520.9538	0.0459
		BA	520.9281	521.0124	520.9659	**0.0287**
		DVBA	**520.00**	**520.1765**	**520.0774**	0.0611
f6	Shifted and Rotated Weierstrass Function	PSO	**618.2712**	630.1718	624.1326	5.1673
		SA	638.511	640.2438	639.2904	**0.5939**
		GA	620.9471	**622.8304**	**622.2639**	0.6728
		BA	628.3503	635.8235	632.2023	3.0088
		DVBA	620.1429	627.6139	623.4401	3.1689
f7	Shifted and Rotated Griewank's Function	PSO	704.6016	721.623	712.0757	4.4171
		SA	1242.3151	1374.9653	1312.3624	44.3206
		GA	701.1195	701.1415	701.1298	**0.0071**
		BA	701.5299	701.6826	701.6131	0.0543
		DVBA	**701.0548**	**701.0956**	**701.0752**	0.0118
f8	Shifted Rastrigin's Function	PSO	800.0004	**800.3548**	800.0765	**0.0861**
		SA	800.0074	800.8517	800.2027	0.1927
		GA	800.0002	805.1329	801.6664	1.6650
		BA	800.0001	801.9901	800.4695	0.5553
		DVBA	**800.0**	800.9949	**800.0331**	0.1785
f9	Shifted and Rotated Rastrigin's Function	PSO	922.8755	942.1456	932.1903	**5.7790**
		SA	940.7263	973.5267	959.8805	10.3234
		GA	914.3183	947.0765	930.1015	8.9776
		BA	950.5951	996.7199	969.1791	15.1363
		DVBA	**907.1656**	**937.5058**	**921.3884**	9.5254
f10	Shifted Schwefel's Function	PSO	**1348.394**	2608.6882	1807.4446	437.264
		SA	2168.5492	2534.9047	2355.6111	**112.8694**
		GA	1440.1039	2388.3572	1998.4405	336.7253
		BA	1354.3348	2626.6369	2025.5035	323.0562
		DVBA	1594.6093	**2093.0453**	**1789.9543**	163.7572
f11	Shifted and Rotated Schwefel's Function	PSO	6872.0808	8722.5079	7062.2627	901.4843
		SA	8381.9828	8569.6215	8326.8159	**266.9547**
		GA	4801.0908	6823.6747	5979.5179	671.5954
		BA	7298.6448	7841.5608	7172.0521	494.7482
		DVBA	**4605.8075**	**6693.6749**	**5032.7305**	523.6241
f12	Shifted and Rotated Katsuura Function	PSO	1202.112	1202.714	1202.523	0.2451
		SA	1202.599	1203.313	1203.023	0.2930
		GA	1200.799	**1200.872**	**1200.836**	**0.0311**
		BA	1201.860	1202.468	1202.119	0.2199
		DVBA	**1200.565**	1201.400	1200.981	0.3085
f13	Shifted and Rotated HappyCat Function	PSO	1300.510	1300.929	1300.668	0.1177
		SA	1305.468	1307.751	1306.823	0.7954
		GA	**1300.153**	**1300.195**	**1300.173**	**0.0117**
		BA	1300.363	1300.565	1300.469	0.0548
		DVBA	1300.374	1300.710	1300.511	0.1037
f14	Shifted and Rotated HGBat Function	PSO	1400.315	1400.622	1400.445	0.0964
		SA	1549.742	1674.734	1627.912	33.204
		GA	1400.182	1400.330	1400.265	0.0387
		BA	1400.203	1400.370	1400.267	0.0441
		DVBA	**1400.178**	**1400.230**	**1400.204**	**0.0205**

Table 5.5: Optimization results for simple multimodal functions-2.

	Function	**Algorithms**	**BFV**	**WFV**	**Mean**	**STDEV**
	Shifted-Rotated	PSO	1503.102	1508.183	1505.913	1.3824
	Expanded	SA	1539.519	2447.317	2024.951	351.127
f15	Griewank'sPlus	GA	**1500.892**	**1501.900**	**1501.522**	**0.3267**
	Rosenbrock's	BA	1501.614	1503.074	1502.644	0.4226
	Function	DVBA	1501.668	1503.441	1502.491	0.6382
		PSO	**1611.374**	1612.965	**1612.349**	0.4810
	Shifted-Rotated	SA	1612.856	1613.382	1613.115	**0.1548**
f16	Expanded	GA	1612.317	1613.482	1613.093	0.3964
	Scaffer'sF6	BA	1612.056	1613.493	1612.498	0.4041
	Function	DVBA	1611.679	**1611.679**	1612.698	0.4228

Table 5.6: Optimization results for hybrid functions.

	Function	**Algorithms**	**BFV**	**WFV**	**Mean**	**STDEV**
		PSO	7.183E+4	7.742E+5	3.767E+5	2.393E+5
	Hybrid	SA	1.559E+5	3.572E+6	2.664E+6	6.014E+5
f17	Function	GA	3.739E+4	1.819E+5	1.110E+5	4.467E+4
	1 (N=3)	BA	6.156E+4	2.626E+5	1.488E+5	6.171E+4
		DVBA	**1.544E+4**	**1.269E+5**	**8.266E+4**	**3.421E+4**
		PSO	1.415E+6	6.898E+6	3.844E+6	1.506E+6
	Hybrid	SA	5.044E+8	2.243E+9	1.277E+9	4.990E+8
f18	Function	GA	4.863E+4	1.031E+5	6.698E+4	1.775E+4
	2 (N=3)	BA	4.265E+5	1.098E+6	7.710E+5	2.089E+5
		DVBA	**2.227E+3**	**2.790E+4**	**9.709E+3**	**9.355E+3**
		PSO	1914.5406	1984.3339	1936.6258	28.042
	Hybrid	SA	2116.2252	2288.9634	2202.7893	55.8765
f19	Function	GA	**1912.7812**	1972.3023	1925.9049	22.8797
	3 (N=4)	BA	1916.6011	1922.9181	1919.2396	**1.6812**
		DVBA	1914.3625	**1922.8812**	**1918.1415**	2.3353
		PSO	5.538E+3	4.4178E+3	2.0795E+4	1.2942E+4
	Hybrid	SA	3.5028E+4	1.3272E+5	7.1581E+4	2.7434E+4
f20	Function	GA	1.5025E+4	5.5645E+4	3.2168E+4	1.2801E+4
	4 (N=4)	BA	2.803E+3	4.393E+3	3.972E+3	4.41E+2
		DVBA	**2.378E+3**	**3.299E+3**	**2.829E+3**	**3.22E+2**
		PSO	1.5115E+5	2.9571E+6	9.6150E+5	7.3422E+5
	Hybrid	SA	2.0810E+6	1.0926E+7	6.2324E+6	2.8236E+6
f21	Function	GA	3.8486E+4	4.4802E+5	1.4765E+5	1.1838E+5
	5 (N=5)	BA	4.9654E+5	1.5662E+6	7.4299E+5	3.0326E+5
		DVBA	**2.9838E+4**	**1.7447E+5**	**8.6698E+4**	**4.2869E+4**
		PSO	**2332.0701**	3682.0521	2988.2845	357.6164
	Hybrid	SA	2960.3060	3908.1976	3562.9155	263.1245
f22	Function	GA	2819.2713	3353.1558	3095.5344	196.6240
	6 (N=5)	BA	2792.5289	3506.6294	3225.1467	215.0238
		DVBA	2405.9625	**2983.6424**	**2670.1793**	**176.2715**

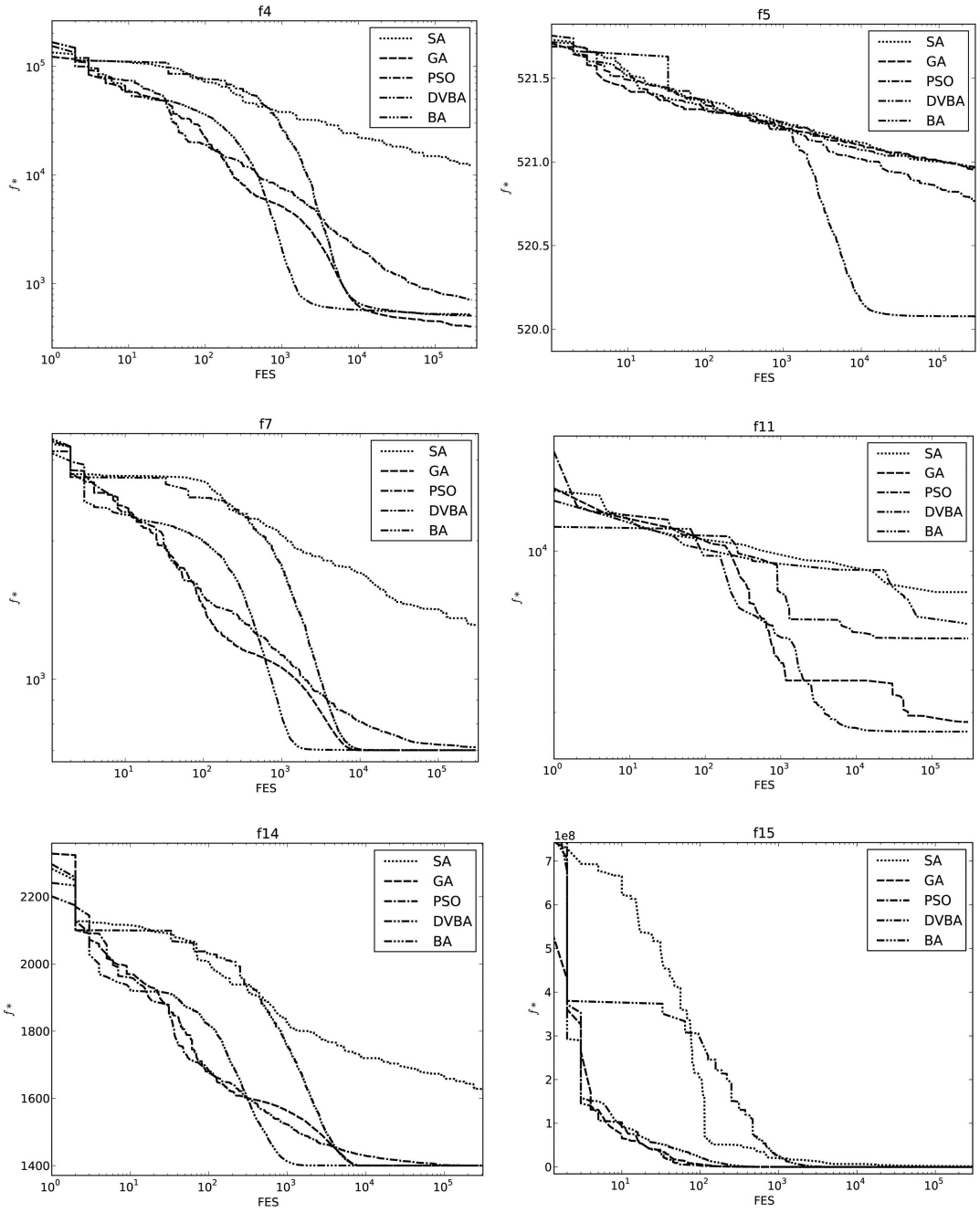


Figure 5.4: Convergence characteristics of SA, GA, PSO, DVBA, and BA on multi-modal functions in CEC 2014.

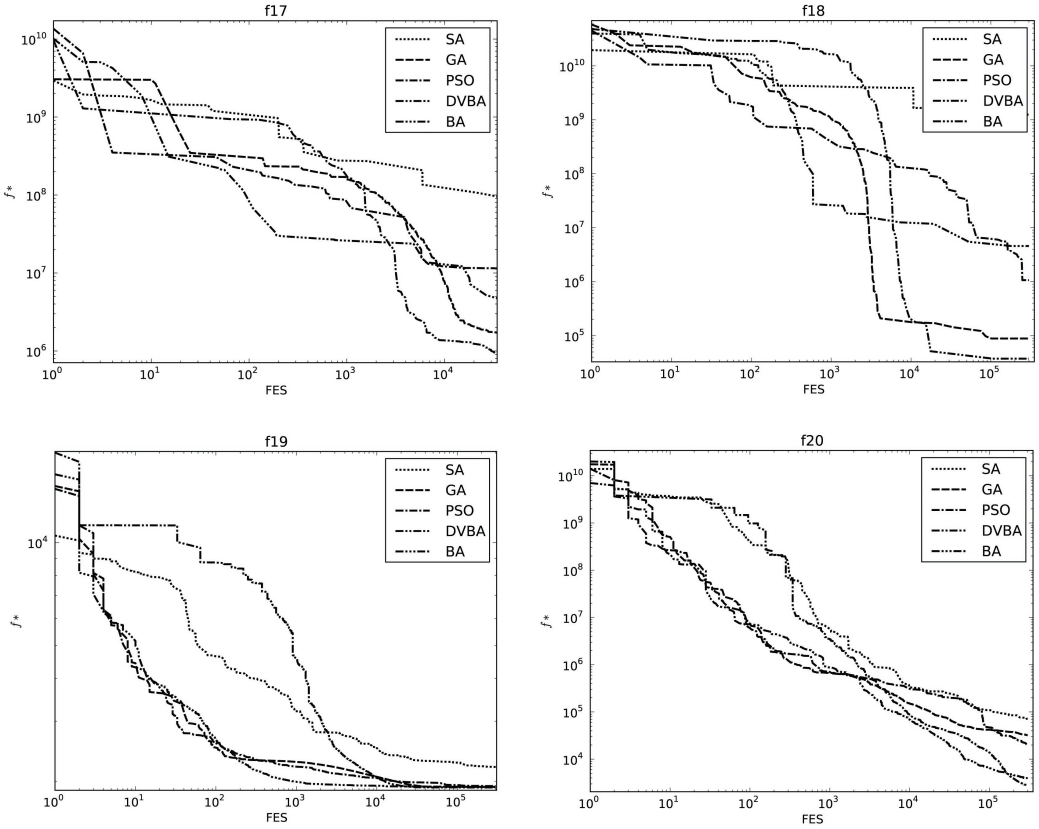


Figure 5.5: Convergence characteristics of SA, GA, PSO, DVBA, and BA on hybrid functions in CEC 2014.

the functions, f_{21} and f_{22} are not shown. For the functions f_{17} and f_{18}, BA starts converging faster but it could not avoid from local optimum traps, but GA and DVBA start slowly the converging and later on successfully find the global optimum without trapping in a local optima (see Fig.5.5). For f_{19}, DVBA, as in general, converges slowly at the beginning, after a learning period its convergence accelerates instantly. BA, GA, and, PSO shows same converging map for f_{19}. All the algorithms demonstrate the same characteristics of convergence for the function f_{20}, especially PSO, BA, and GA.

Empirical results on composition functions from CEC 2014

In this group, there are 8 composition functions. The composition function merges the properties of the sub-functions better and maintains continuity around the global/local optima. The details of the functions can be found in [55]. For the functions f_{25}, f_{26}, f_{27}, and f_{28}, DVBA has all the best Mean values compared with other algorithms. For f_{23}, f_{29}, and f_{30}, GA performed better than all other algorithms followed by DVBA with small difference. Specially, for function f_{29}, GA and DVBA outperformed significantly other algorithms in terms of Mean. Similarly, for function f_{24}, DVBA has the second best solution, which is significantly better than SA, GA, and BA. PSO has the best solution. %50 of the test functions in this group, DVBA has the best solution in terms of Mean. We can say that, DVBA is also highly competitive optimization algorithm for solving these Composition functions.

Summary

Table 5.8 and Table 5.9 summarize the aggregate accuracy of each algorithm. In Table 5.8, for each function groups, the overall success rate of algorithms in terms of Mean is shown. If an algorithm has the best Mean value for a function, it is called success for the algorithm. The overall success rate is calculated as the number of the successes divided by total number of functions in the group. Same method is applied for BFV, WFV, Mean, and STDEV to calculate overall success rate for each algorithm on 36 functions without group distribution in Table 5.9.

In Table 5.8, it can be seen that, DVBA had the best Mean values for 63.9% of 36 functions. Specially, DVBA outperformed the rest four algorithms for all Hybrid functions and Unimodal (shifted and Rotated) functions. GA has the second best performance, followed by PSO, BA, then SA.

According to Table 5.8 and Table 5.9, GA remained the toughest competitors of DVBA in most of the cases. For Multimodal (Shifted and Rotated) functions, while DVBA was able to get the best results for more than half of the functions (53.9%), GA got the best results for

Table 5.7: Optimization results for composition functions.

	Function	Algorithms	BFV	WFV	Mean	STDEV
		PSO	2627.1944	2647.4335	2637.5313	5.9261
	Composition	SA	2838.6484	3143.5463	3036.2873	82.2439
f23	function	GA	**2615.4249**	**2615.5528**	**2615.4938**	**0.0366**
	1 (N=5)	BA	2616.0486	2617.6914	2616.6271	0.5205
		DVBA	2615.9433	2617.0687	2616.3877	0.3609
		PSO	**2600.1725**	**2601.0021**	**2600.6348**	**0.2414**
	Composition	SA	2782.8495	2831.0690	2808.2579	17.8240
f24	function	GA	2647.0285	2688.3520	2659.6338	11.5622
	2 (N=5)	BA	2673.9840	2681.6120	2677.6541	2.3091
		DVBA	2626.3381	2665.6850	2649.4845	10.4219
		PSO	2713.7340	**2726.4403**	2720.8272	**3.6117**
	Composition	SA	2754.2442	2786.3128	2770.0326	10.4771
f25	function	GA	**2705.0392**	2737.3801	2721.4788	9.5331
	3 (N=5)	BA	2705.8571	2796.2565	2728.3825	27.3140
		DVBA	2706.8784	2737.1816	**2718.7843**	9.1166
		PSO	**2700.2118**	2700.7442	2700.3105	0.1463
	Composition	SA	2701.0920	2702.0005	2701.6031	0.32
f26	function	GA	2700.0557	2731.1702	2704.0389	9.202
	4 (N=5)	BA	2700.1165	2700.5512	2700.2920	0.1307
		DVBA	2700.159	**2700.3926**	**2700.2553**	**0.0753**
		PSO	3128.1967	4401.6944	3918.4691	430.7498
	Composition	SA	3393.0577	3883.0382	3623.3018	**138.1861**
f27	function	GA	**3101.1794**	**3682.7039**	3502.0625	208.775
	5 (N=5)	BA	3153.4002	3831.2028	3571.5564	270.9837
		DVBA	3103.7064	3719.8534	**3274.7901**	261.5365
		PSO	4052.9321	5517.2783	4784.3893	512.2478
	Composition	SA	7053.41	8350.2419	7885.7531	387.8475
f28	function	GA	5252.6716	7884.4142	6481.6061	776.8109
	6 (N=5)	BA	4455.4575	6447.3395	5233.2734	552.4140
		DVBA	**3870.2498**	**4997.7464**	**4366.7638**	**308.6143**
		PSO	3.843E+4	1.613E+8	8.132E+7	1.131E+7
	Composition	SA	1.163E+8	2.472E+8	1.872E+8	**1.731E+4**
f29	function	GA	**9.898E+3**	**2.1158E+4**	**1.654E+4**	2.361E+5
	7 (N=5)	BA	3.544E+5	4.1924E+6	2.001E+6	1.323E+6
		DVBA	1.484E+4	2.7884E+4	2.134E+4	2.207E+4
		PSO	2.193E+4	1.415E+6	4.626E+5	4.934E+4
	Composition	SA	2.815E+5	2.082E+6	1.017E+6	5.003E+5
f30	function	GA	**5.578E+3**	**9.795E+3**	**7.119E+3**	**1.317E+3**
	8 (N=5)	BA	2.091E+4	1.382E+5	6.522E+4	3.854E+4
		DVBA	9.571E+3	2.657E+4	1.494E+4	5.091E+3

Table 5.8: Comparison of Accuracy of Algorithms according to the test function groups in CEC 2014 test suit (OSM).

Functions	**# of Functions**	**PSO**	**SA**	**GA**	**BA**	**DVBA**
Classic Test Functions	6	0.0%	0.0%	33.3%	16.6%	**50.0%**
Unimodal(Shifted-Rotated) Functions	3	0.0%	0.0%	0.0%	0.0%	**100.0%**
Multimodal (Shifted-Rotated) Functions	13	7.69%	0.0%	38.5%	0.0%	**53.9%**
Hybrid Functions	6	0.0%	0.0%	0.0%	0.0%	**100.0%**
Composition Functions	8	12.5%	0.0%	37.5%	0.0%	**50.0%**
Overall Results	36	5.5%	0.0%	27.8%	2.8%	**63.9%**

Table 5.9: Comparison of accuracy of the tested optimization algorithms.

% of	**PSO**	**SA**	**GA**	**BA**	**DVBA**
BFV	16.8%	0.0%	30.6%	0.0%	**52.8%**
WFV	8.3%	0.0%	30.6%	2.8%	**58.3%**
Mean	5.5%	0.0%	27.8%	2.8%	**63.9%**
STDEV	13.9%	16.7%	**27.8%**	13.9%	**27.8%**

38.5% of functions.

In Table 8, it is seen that, DVBA outperformed significantly for most of the functions in terms of BFV, WFV, and Mean. Only GA had the same overall success (OS) ratio with DVBA in terms of STDEV (see Table 5.9).

A close inspection of Table 5.2, 5.3, 5.4, 5.5, 5.6, and 5.7 reveals that, out of 36 functions, DVBA outperformed all other contestant algorithms on 23 functions in terms of Mean. It can be clearly seen that DVBA can provide a better optimization solution for most of the functions encompassing different type of functions with low and high dimensions. Therefore, it can be said that DVBA has the best universality on different type of problems.

5.3.4 Comparison the algorithms in Group 2 and 3

The performances of the algorithms in the second and the third group are extensively evaluated on a suite of 30 bound-constrained optimization problems from CEC 2014. According to the instructions in CEC 2014 special session we set the maximum number of FEs 3×10^5 for 30-D problems and 1×10^6 for 100-D problems. For the algorithms in group 2, the dimensions of the functions are set to 30 and the results are shown in Table 5.10. To compare the performance of DVBA for high dimensional optimization problems, the results of the algorithms are shown for

100-D test functions in Table 5.11. In this group, the numerical benchmark results were taken from the aforementioned papers. In the table 5.10 and 5.11, the test results are shown in terms of the mean error (MeanErr) and the standard deviation (STDEV) of the results. The mean error values are found according to $(F(x) - F(x^*))$ for evaluating the success of algorithms, where x is the best found value in a run and x^* is the global best of the test function. The best mean error values are typed in bold.

Furthermore, we used two-tailed t-tests [14] to compare the mean error values of the results produced by the DVBA and the other algorithms at the 0.05 level of significance. The statistical significance level of the aggregate results are shown at the last three rows of the Table 5.10 and Table 5.11. There '+' indicates that a given algorithm performed significantly more successful than DVBA, '−' means DVBA was better than given algorithm, and '≈' indicates that DVBA and compared algorithm are not significantly different better or worse. In addition, Table 5.12 shows the aggregate results of comparing each algorithm from group 3 vs. DVBA using the Wilcoxon rank-sum test [101].

Comparison of DVBA with four state-of-the-art BA variants

It can be seen from Table 5.10 that BA variants do not show significant difference for unimodal functions (f_1 and f_2). Except for function f_3, the solutions with DVBA are significantly better than with other algorithms. For multi-modal functions (f_4-f_{16}), DVBA has all the best mean error values compared with other algorithms. For function f_{13}, NABA and CLSBA have the smallest mean error values which are not significantly inferior to DVBA. Similarly, for function f_4, LMSBA finds a competitive solution when compared with DVBA, as its t-test value reflects. However, LMSBA and ABA shows better performance than DVBA for function f_{13}.

For functions f_{17}, f_{18}, and f_{22}, DVBA is significantly superior to the other four algorithms. However, DVBA has the worst mean error value for function f_{20}. This is partly because the search scope size of the explorer bat is not vast enough to escape from a large local optima trap. For the hybrid functions, DVBA has the best mean error values, followed by LMSBA and NABA. Similarly, for composition functions, DVBA shows better performance than the rest

four BA variants. For function f_{20}, although LMSBA has the smallest mean error, followed by ABA, the algorithms did not show significant difference in terms of the mean error values.

According to t-test results, algorithms are significantly better than DVBA for functions f_3 and f_{20}. For functions f_{13} and f_{24}, the performance of DVBA was not better than other algorithms. However, DVBA outperformed the modified BA algorithms for the rest of the functions. Overall, we can say that, DVBA is a highly competitive algorithm for solving high-dimensional test functions compared with BA variants.

Comparison of DVBA with five state-of-the-art algorithms competed in special session at CEC 2014

In this section, we intend to show how well the proposed DVBA algorithm performs when compared to FWA-DM, L-SHADE, b636rl, NRGA, and OPTBees which are lately competed in CEC 2014 on Single Objective Real Parameter Numerical Optimization Competition. Table 5.11 reports the mean error and standard deviation of function values by applying seven algorithms to optimize the 100-D CEC 2014 numerical functions. The aggregate results of statistical testing ($+$, $-$, $\approx$) on 30 functions are shown at the last three rows of Table 5.11.

As Table 5.11 shows, for function f_1, all six algorithms did not find its global optimum. By analyzing their t-test values, we can see that the solutions with DVBA are significantly better than that of other algorithms. Similarly, for functions f_{11}, f_{16}, and f_{22}, DVBA is significantly better than other algorithms. However, no significant difference was observed on comparison with the remaining five algorithms. The performance of the algorithms on rotated and un-rotated functions can be clearly seen on functions $f_8 - f_9$ and $f_{10} - f_{11}$. While the function f_8 is shifted Rastrigin, f_9 is shifted and rotated Rastrigin's function. Similarly, f_{10} is shifted Schwefel's function and f_{11} is shifted and rotated Schwefel's function. As can be seen from the results for f_8 and f_9, DVBA gives better results for un-rotated ones. Same results observed for functions f_{10} and f_{11}.

In addition to an overall comparison, DVBA compared with these algorithms on specific sub-classes of problems in CEC 2014 benchmark set. The results on these sub-classes are

shown in Table 5.12. The table shows the aggregate results of comparing each algorithm vs. DVBA using the Wilcoxon rank-sum test (significantly, $p < 0.05$)

On 3 unimodal functions f_1, f_2, and f_3, DVBA performs better than FWA-DM and NRGA and similarly to b636rl, but DVBA is outperformed by both L-SHADE and OPTBees. For 13 simple, multimodal functions, DVBA clearly outperforms other algorithms, but L-SHADE performs better than DVBA. For 6 hybrid functions, while DVBA outperforms FWA-DM, NRGA, OPTBees, and b636rl, it is outperformed by L-SHADE. For 8 composition functions, except NRGA, DVBA is outperformed by rest of the algorithms. Specially, for functions f_{29} and f_{30}, DVBA got stuck in local solutions.

We can say that, DVBA has good solutions for multimodal functions ($f_4 - f_{16}$) and hybrid functions ($f_{17} - f_{22}$), but performs sub-optimal for composition functions compared with state-of-the-art algorithms.

In short, it can be seen from Tables 5.10, 5.11, and 5.12 that the algorithms in this group are more competitive than other algorithms. DVBA performs better than FWA-DM, NRGA, and b636rl while it exhibits similar performance with OptBees. For functions f_1, f_5, f_{11}, f_{16}, f_{19}, f_{22}, and f_{24}, DVBA finds better solutions than L-SHADE. However, for the rest of the functions, L-SHADE finds better or competitive solutions compared with DVBA, as their t-test values and the Wilcoxon rank-sum test reflect.

Table 5.10: Comparison of LMSBA, NABA, ABA, CLSBA, and DVBA over 30 test functions of 30 dimensions using 300,000 function evaluations."MeanErr" and "StdDev" indicate the mean error and standard deviation of the results found over the 30 independent runs by each algorithm.

Func.	LMSBA		NABA		ABA		CLSBA		DVBA
	MeanErr (StdDev)	t-test	MeanErr (StdDev)	t-test	MeanErr (StdDev)	t-test	MeanErr (StdDev)	t-test	MeanErr (StdDev)
f_1	6.26E+06(2.25E+06)	−	6.17E+06(1.20E+06)	−	7.30E+06(2.21E+06)	−	6.40E+06(1.22E+06)	−	**1.05E+05**(4.28E+05)
f_2	7.50E+07(2.61E+06)	−	5.74E+07(2.92E+06)	−	7.08E+07(4.93E+06)	−	6.53E+07(6.56E+06)	−	**2.91E+04**(1.32E+04)
f_3	1.36E+03(5.39E+02)	+	**2.48E+02**(2.73E+01)	+	1.54E+03(8.97E+02)	+	5.20E+02(1.02E+02)	+	1.59E+04(6.25E+03)
f_4	**1.03E+02**(4.69E+01)	≈	1.25E+02(4.03E+01)	−	1.30E+02(3.39E+01)	−	1.61E+02(7.99E+01)	−	1.08E+02(3.31E+01)
f_5	2.07E+01(4.02E-02)	−	2.08E+01(2.99E-02)	−	2.10E+01(3.02E-02)	−	2.10E+01(3.02E-02)	−	**2.01E+01**(6.11E-02)
f_6	2.80E+01(4.52E+00)	−	2.94E+01(3.14E+00)	−	2.85E+01(1.15E+00)	−	2.94E+01(3.27E+00)	−	**2.34E+01**(3.16E+00)
f_7	1.60E+00(9.00E-02)	−	1.48E+00(4.43E-02)	−	1.58E+00(6.60E-02)	−	1.54E+00(9.95E-02)	−	**1.07E+00**(1.12E-02)
f_8	2.10E-01(2.78E-01)	−	2.40E-01(2.51E-01)	−	3.20E-01(5.51E+01)	−	2.60E-01(5.43E-01)	−	**3.00E-02**(1.76E-01)
f_9	5.70E+01(5.13E+01)	−	5.80E+01(6.01E+01)	−	6.00E+01(5.32E+01)	−	4.60E+01(3.03E+01)	−	**2.14E+01**(9.52E+00)
f_{10}	1.49E+03(6.68E+02)	−	1.04E+03(5.99E+02)	−	1.73E+03(5.72E+02)	−	1.69E+03(6.34E+02)	−	**7.90E+02**(1.63E+02)
f_{11}	4.97E+03(3.80E+02)	−	5.26E+03(3.57E+02)	−	5.20E+03(6.37E+02)	−	5.14E+03(6.10E+02)	−	**3.93E+03**(5.23E+02)
f_{12}	2.08E+00(2.12E-01)	−	1.93E+00(1.54E-01)	−	1.87E+00(3.16E-01)	−	2.03E+00(1.94E-01)	−	**9.80E-01**(3.07E-01)
f_{13}	**3.90E-01**(2.69E-02)	+	4.90E-01(1.27E-01)	≈	4.30E-01(6.11E-02)	+	4.90E-01(7.33E-02)	≈	5.10E-01(1.03E-01)
f_{14}	2.70E-01(3.62E-02)	−	2.60E-01(3.74E-02)	−	2.90E-01(4.71E-02)	−	2.70E-01(5.46E-02)	−	**2.00E-01**(2.05E-02)
f_{15}	1.75E+01(1.45E+00)	−	5.87E+00(1.26E+00)	−	2.11E+01(6.11E-01)	−	1.86E+01(1.16E+00)	−	**2.49E+00**(6.33E-01)
f_{16}	1.36E+01(1.85E-01)	−	1.33E+01(4.86E-01)	−	1.30E+01(7.35E-01)	−	1.34E+01(3.58E-01)	−	**1.27E+01**(4.27E-01)
f_{17}	1.23E+05(2.39E+05)	−	1.26E+05(6.67E+04)	−	1.70E+05(3.09E+05)	−	1.29E+05(1.84E+05)	−	**8.09E+04**(3.42E+04)
f_{18}	1.01E+05(3.43E+05)	−	4.15E+05(2.09E+05)	−	2.17E+05(1.36E+05)	−	1.04E+05(1.17E+04)	−	**7.90E+03**(9.35E+03)
f_{19}	1.49E+01(1.86E+00)	≈	**1.41E+01**(1.97E+00)	+	2.80E+01(2.71E+01)	−	1.49E+01(7.02E-01)	≈	1.81E+01(2.33E+00)
f_{20}	**3.50E+02**(5.09E+01)	+	3.87E+02(1.16E+02)	+	4.06E+02(1.07E+02)	+	5.30E+02(5.34E+01)	+	8.29E+02(3.22E+02)
f_{21}	8.34E+04(6.63E+04)	≈	**7.57E+04**(3.51E+04)	≈	1.36E+05(8.29E+04)	−	1.13E+05(6.87E+04)	−	8.45E+04(4.28E+04)
f_{22}	9.00E+02(1.60E+02)	−	1.00E+03(2.59E+02)	−	1.01E+03(2.06E+02)	−	1.03E+03(8.22E+01)	−	**4.70E+02**(1.76E+02)
f_{23}	3.17E+02(5.08E-01)	−	3.17E+02(3.63E-01)	−	3.19E+02(2.49E+00)	−	3.17E+02(3.74E-01)	−	**3.16E+02**(3.61E-01)
f_{24}	**2.35E+02**(6.64E+00)	+	2.43E+02(1.31E+01)	≈	2.39E+02(7.93E+00)	+	2.55E+02(3.28E+01)	≈	2.49E+02(1.04E+01)
f_{25}	2.28E+02(1.09E+01)	−	**2.17E+02**(1.96E+01)	≈	2.29E+02(3.73E+01)	≈	2.19E+02(1.13E+01)	≈	2.19E+02(9.11E+00)
f_{26}	1.01E+02(5.70E-02)	−	1.20E+02(4.01E+01)	−	**1.00E+02**(9.34E-02)	−	1.41E+02(4.90E+01)	−	**1.00E+02**(7.53E-02)
f_{27}	**5.40E+02**(2.83E+02)	≈	1.35E+03(9.83E+01)	−	1.06E+03(3.40E+02)	−	1.25E+03(9.83E+01)	−	5.70E+02(2.61E+02)
f_{28}	2.10E+03(7.58E+02)	−	2.52E+03(6.41E+02)	−	2.50E+03(1.12E+02)	−	3.78E+03(3.81E+02)	−	**1.56E+03**(3.08E+02)
f_{29}	2.54E+06(4.99E+06)	−	6.71E+06(9.27E+06)	−	4.06E+06(5.06E+06)	−	1.87E+06(3.63E+06)	−	**1.84E+04**(2.20E+04)
f_{30}	**4.50E+03**(5.74E+02)	−	1.11E+04(4.67E+03)	≈	1.48E+04(1.73E+03)	−	1.19E+04(8.17E+03)	≈	1.19E+04(5.09E+03)
+	4		3		4		2		
−	22		22		25		23		
≈	4		5		1		5		

"+", "−", and "≈" denote that the performance of the corresponding algorithm is significantly better than, worse than, and significantly not different to that of DVBA, respectively.

Table 5.11: Comparison of DVBA with FWA-DM, L-SHADE, NRGA, OPTBees, and b6e6rl over 30 test functions of 100 dimensions using 1,000,000 function evaluations. "MeanErr" and "StdDev" indicate the mean error and standard deviation of the results found over the 51 independent runs by each algorithm.

Func.	FWA-DM MeanErr (StdDev)	L-SHADE MeanErr (StdDev)	NRGA MeanErr (StdDev)	OPTBees MeanErr (StdDev)	b6e6rl MeanErr (StdDev)	DVBA MeanErr (StdDev)
f_1	2.28E+08 (4.08E+07)	1.70E+05 (5.70E+04)	3.24E+07 (4.56E+06)	3.00E+05 (1.00E+05)	1.31E+06 (4.07E+05)	**1.25E+05**(1.35E+04)
f_2	1.62E+04 (1.81E+04)	**0.00E+00** (0.00E+00)	1.46E+04 (6.68E+03)	1.09E+01 (2.92E+01)	2.22E+04 (2.69E+04)	3.87E+04(1.97E+06)
f_3	2.95E+04 (3.64E+03)	**0.00E+00** (0.00E+00)	2.70E+04 (4.06E+03)	7.50E+02 (7.86E+02)	6.94E+02 (7.93E+02)	1.59E+04(2.36E+02)
f_4	1.83E+02 (1.00E+02)	1.70E+02 (3.10E+01)	3.96E+02 (3.51E+01)	**1.42E+02** (5.48E+01)	1.74E+02 (3.34E+01)	1.51E+02(3.43E+01)
f_5	2.10E+01 (2.44E-02)	2.10E+01 (3.10E-02)	**2.00E+01** (8.09E-05)	**2.00E+01** (2.49E-05)	2.06E+01 (2.30E-02)	2.03E+01(2.20E-02)
f_6	1.14E+02 (2.54E+00)	8.70E+00 (2.30E+00)	9.76E+01 (4.93E+00)	7.15E+01 (7.73E+00)	7.40E+01 (2.98E+00)	2.86E+01(6.81E+00)
f_7	1.29E-01 (3.14E-02)	**0.00E+00** (0.00E+00)	2.22E-02 (7.34E-03)	5.89E-03 (8.05E-03)	5.68E-08 (1.09E-08)	1.18E+00(1.73E-01)
f_8	1.08E+02 (5.40E+00)	1.10E-02 (7.40E-03)	2.00E+02 (2.39E+01)	**1.42E-12** (3.05E-13)	1.08E-08 (5.06E-09)	1.20E-01(2.82E+02)
f_9	5.52E+02 (4.44E+01)	**3.40E+01** (5.00E+00)	2.45E+02 (2.22E+01)	6.66E+02 (9.79E+01)	3.21E+02 (3.44E+01)	9.84E+01(2.54E+02)
f_{10}	5.67E+03 (3.04E+02)	2.60E+01 (5.80E+00)	6.33E+03 (1.28E+03)	4.26E+03 (4.30E+02)	**5.30E-02** (4.50E-02)	8.42E+02(1.38E+03)
f_{11}	1.46E+04 (6.30E+02)	1.10E+04 (5.60E+02)	1.37E+04 (1.56E+03)	1.24E+04 (1.11E+03)	1.18E+04 (4.80E+02)	**8.67E+03**(1.06E+03)
f_{12}	1.21E+00 (7.78E-02)	4.40E-01 (4.70E-02)	3.80E-01 (8.91E-02)	**2.32E-01** (5.08E-02)	5.04E-01 (3.20E-02)	1.57E+00(2.81E-01)
f_{13}	5.61E-01 (3.61E-02)	**2.40E-01** (2.10E-02)	5.01E-01 (3.00E-02)	5.90E-01 (8.16E-02)	5.23E-01 (5.20E-02)	5.84E-01(7.78E-02)
f_{14}	1.89E-01 (2.06E-02)	**1.20E-01** (7.30E-03)	1.63E-01 (7.21E-03)	2.27E-01 (2.27E-02)	2.18E-01 (1.60E-02)	2.21E-01(2.61E-02)
f_{15}	8.74E+01 (5.87E+00)	**1.60E+01** (1.20E+00)	4.53E+02 (5.23E+01)	6.59E+01 (1.83E+01)	4.09E+01 (4.47E+00)	2.81E+01(8.01E+00)
f_{16}	4.35E+01 (3.75E-01)	3.90E+01 (4.80E-01)	4.36E+01 (1.04E+00)	4.09E+01 (1.17E+00)	3.97E+01 (6.54E-01)	**3.15E+01**(3.03E-01)
f_{17}	2.31E+07 (5.63E+06)	**4.40E+03** (7.10E+02)	2.17E+06 (4.83E+05)	1.09E+05 (6.74E+04)	1.86E+05 (5.31E+04)	9.19E+04(1.57E+05)
f_{18}	5.68E+03 (8.70E+03)	**2.20E+02** (1.70E+01)	6.32E+02 (3.11E+02)	1.59E+03 (2.11E+03)	9.41E+02 (8.65E+02)	1.64E+04(1.18E+06)
f_{19}	6.34E+01 (2.43E+00)	9.60E+01 (2.30E+00)	9.93E+01 (1.97E+01)	**5.28E+01** (1.56E+01)	9.47E+01 (1.20E+01)	8.26E+01(2.18E+01)
f_{20}	6.93E+04 (1.10E+04)	**1.50E+02** (5.20E+01)	7.17E+04 (1.42E+04)	1.06E+04 (4.24E+03)	7.73E+03 (3.38E+03)	1.18E+03(1.45E+02)
f_{21}	9.57E+06 (2.31E+06)	**2.30E+03** (5.30E+02)	1.92E+06 (4.77E+05)	3.11E+05 (1.60E+05)	8.92E+04 (3.92E+04)	2.23E+05(1.03E+06)
f_{22}	1.51E+03 (1.34E+02)	1.10E+03 (1.90E+02)	2.29E+03 (4.53E+02)	2.03E+03 (3.31E+02)	1.88E+03 (2.31E+02)	**8.94E+02**(4.67E+02)
f_{23}	3.46E+02 (2.18E-01)	3.50E+02 (2.80E-13)	3.70E+02 (3.30E+00)	**3.46E+02** (9.28E-01)	3.48E+02 (2.24E-08)	3.64E+02(7.66E-01)
f_{24}	3.63E+02 (2.85E+00)	3.90E+02 (2.90E+00)	3.76E+02 (4.32E+00)	**3.49E+02** (1.05E+01)	3.63E+02 (2.38E+00)	3.85E+02(1.02E+01)
f_{25}	3.03E+02 (1.74E+01)	**2.00E+02** (4.00E-13)	2.27E+02 (2.07E+01)	2.08E+02 (1.12E+00)	2.48E+02 (9.05E+00)	2.46E+02(4.35E+01)
f_{26}	1.61E+02 (5.88E+01)	2.00E+02 (6.20E-13)	2.00E+02 (4.18E-02)	**1.01E+02** (6.84E-02)	2.00E+02 (4.40E-02)	2.00E+02(4.19E-01)
f_{27}	3.14E+03 (4.13E+02)	3.80E+02 (3.30E+01)	2.35E+03 (1.34E+02)	2.16E+03 (1.67E+02)	2.02E+03 (1.90E+02)	**1.57E+03**(1.90E+02)
f_{28}	**1.60E+03** (3.42E+02)	2.30E+03 (4.60E+01)	1.21E+04 (8.95E+02)	6.14E+02 (4.04E+01)	3.05E+03 (7.68E+02)	2.15E+03(1.89E+03)
f_{29}	**2.70E+02** (5.23E+00)	8.00E+02 (7.60E+01)	3.81E+03 (4.67E+02)	2.75E+02 (3.27E+00)	1.61E+03 (1.80E+02)	3.52E+04(2.77E+05)
f_{30}	**2.23E+03** (1.16E+03)	8.30E+03 (9.60E+02)	8.73E+04 (1.92E+04)	2.86E+03 (2.42E+00)	8.54E+03 (1.17E+03)	2.87E+05(1.31E+05)
$+$	10	17	6	12	11	
$-$	16	7	18	12	14	
$\approx$	4	6	6	6	5	

"+", "−", and "≈" denote that the performance of the corresponding algorithm is significantly better than, worse than, and significantly not different to that of DVBA, respectively.

Table 5.12: Comparison of DVBA with FWA-DM, L-SHADE, NRGA, OPTBees, and b6e6rl on the CEC 2014 benchmarks for 100-D on 4 groups. "+", "−", and "≈" denote that a given algorithm performed significantly better (+), significantly worse (−), or not significantly different (≈) compared to DVBA using the Wilcoxon rank-sum test. All results based on 51 independent runs.

Groups	vs. DVBA (Wilcoxon rank-sum, p <0.05)	FWA-DM	L-SHADE	NRGA	OPTBees	b6e6rl
3 Unimodal Functions	+ (better)	0	2	0	2	1
	− (worse)	2	1	2	1	1
	≈ (not signi.)	1	0	1	0	1
13 Simple Multimodal Functions	+ (better)	4	7	4	3	5
	− (worse)	8	3	8	6	7
	≈ (not signi.)	1	3	1	4	1
6 Hybrid Functions	+ (better)	2	4	1	2	2
	− (worse)	4	2	5	4	4
	≈ (not signi.)	0	0	0	0	0
8 Composition Functions	+ (better)	4	4	2	5	3
	− (worse)	2	1	3	1	2
	≈ (not signi.)	2	3	3	2	3

5.4 Supply Chain Cost Problem

In this section, DVBA has been applied to minimize the supply chain cost with other well-known algorithms; Particle Swarm Optimization (PSO), Bat Algorithm (BA), Genetic Algorithm (GA), and Tabu Search (TS). Optimization of supply chain is considered as a real challenge by researchers because of its complexity. Big number of parameters to be controlled and their distributions, interconnections between parameters and dynamism are the main factors that increase the complexity of a supply chain. The result of the case study showed that the DVBA is much superior to other algorithms in terms of accuracy and efficiency.

A supply chain is an interconnected system of globally distributed business entities that produces and delivers product/services to the customer (buyer). Business entities that compose a supply chain are: suppliers, manufacturers, warehouses, retailers and customers. Suppliers are the provider of raw material. Manufacturers transform the raw materials into final products. Warehouses distribute the products to retailers which in turn sell the products to the ultimate customer [63]. Minimization of cost and maximization of profits for each business entities have emerged a new optimization problem known as Supply Chain Cost Problem

which minimizes the cost of a globally distributed supply chain and maximizes the profits of chain stakeholders as a result [77]. Finding the minimal cost of a supply chain can be solved using traditional deterministic approaches for small instances; however the problem has an exponential complexity growth as the instances become larger and larger. Researchers [5] [7] have proved that supply chain cost problem lies in NP-hard category.

The main entities of a supply chain as shown in Fig.5.6 are: suppliers, manufacturers, warehouses, retailers, and customers. Suppliers provide the raw materials to the manufactures which in turn convert the raw materials to final products. The chain continues then with warehouses, warehouses deliver the products from manufacturers to retailers, and retailers sell these products to the final customers [63].

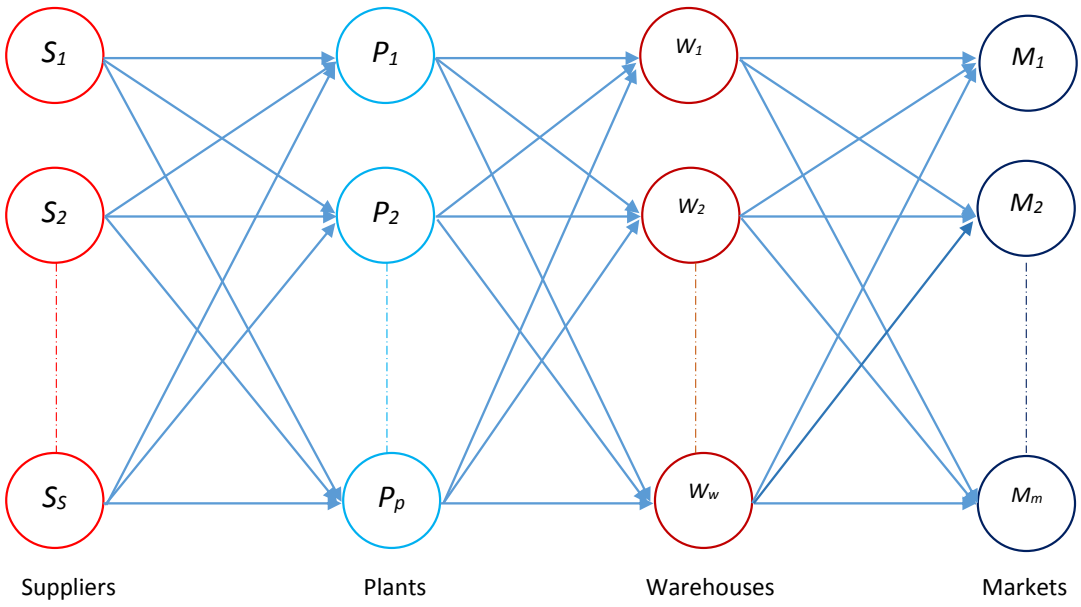


Figure 5.6: Supply Chain

Nowadays, researchers [34] have been focused on the design, analysis and management of supply chain in order to maximize the profits of every stakeholder, minimize the total cost and fulfill customers' need and satisfaction. The total cost equation (5.3) of a globally distributed

supply chain [50] is made up of supply cost of raw material (SCRM), cost of production (PC), cost associated with warehouses (WAC) and cost of markets (MC).

$$TotalCost = SCRM + PC + WAC + MC \tag{5.3}$$

Although equation 5.3 looks simple, researchers are facing a real challenge, because finding the minimal cost of a supply chain is a NP-hard problem [7]. Lately metaheuristic algorithms are used widely for optimizing NP-hard problems [63], since they are simple, easy to be implemented and generally produce acceptable solutions for a wide range of optimization problem.

Table 5.13: Suplly chain cost problem scenarios.

Scenarios	**Suppliers**	**Plants**	**Warehouses**	**Markets**
1	1000 1000	600 400 400	400 300 500 200	100 100 200 70 30
2	500 500 1000	600 400 400	400 300 250 250 200	100 50 50 200 70 30
3	500 500 1000	600 200 200 400	200 200 300 250 250 200	100 50 50 200 70 30
4	500 500 750 250	300 300 200 200 100 300	200 200 300 250 250 200	100 50 50 200 70 30
5	500 500 250 250 250 250	300 300 200 200 100 150 150	200 200 300 250 250 200	100 50 50 200 70 30

5.4.1 Experiments

Algorithms tested with five different supply chain scenarios. In every scenario, demand of each market, capacity of warehouses, production capacity of plants and production capacity of suppliers are created randomly but increased their complexity. Scenarios are schematically represented in Table 5.13. In scenario 1: there are 2 suppliers, s1 and s2, whose production capacities are 1000 and 1000 units; 3 plants (p1, p2, p3) whose production capacities are 600, 400, and 400 units; 4 warehouses (w1, w2, w3, w4) whose storage capacities are 400, 300, 500 and 200 units; and 5 markets(m1, m2, m3, m4, m5) whose demands are 100, 100, 200, 70, and 30 units respectively.

5.4.2 Algorithms for comparison

In order to demonstrate the effectiveness of DVBA we have compared it with PSO, GA, TS, and BA. Each scenario was optimized by each algorithms 30 times. Maximum number of function evaluations (FES) is set to 1000. Furthermore, we set the population size at 30 for BA, GA, and PSO but we limited DVBA's population size at 2. For PSO, we have used the standard PSO and set the inertia weight $w = 1$ and the acceleration coefficients $c_1 = c_2 = 2$ [47]. As for the BA, parameters are set as follows: $\alpha = \gamma = 0.5$, frequency is in the range [0 ,2], and$A_0 = 0.5$. For genetic algorithms GA, we have used the mutation probability of $p_m = 0.05$ and crossover probability of 0.95.

5.4.3 Experimental results and discussions

Table 5.14 reports the best cost value (BCV), the worst cost value (WCV), the mean, the standard deviation (STDEV) of the scenarios, and the mean time spent per trial in seconds. For each scenario the dimensions of the problem is shown in Table 5.14 as well. From Table 5.14, it can be seen that when the dimensions get higher, optimizing the cost is getting more complicated and time consuming. When we check the Mean of the algorithms, DVBA got the smallest mean except the Scenario 3. Although BA, PSO, and GA start the optimization with

30 particles, they could not get better cost then DVBA in general.

From Fig. 5.7, the convergence graphics of PSO, BA, GA, TS, and DVBA shows that DVBA finds better cost then other algorithms in all scenarios. Only in scenario 3, Tabu Search performed slightly better than DVBA; however, in other scenarios Tabu Search did not show the same performance. Because of its complexity, after a few assessments time, algorithms reach their global optimum and then they trap there. However, DVBA manages the exploration better than other algorithms and converges in a better global optimum. In Fig.5.7d shows that, even the dimensions get very high which is 120 D in this scenario, DVBA is much superior to other algorithms in terms of accuracy and efficiency.

Table 5.14: Comparison of PSO, BA, GA, TS, and DVBA on 5 different supply chain cost problem scenarios

Sce.	**Dim.**	**Alg.**	**BCV**	**WCV**	**Mean**	**STDEV**	**Time(sec)**
		PSO	173621.46	313406.01	246699.61	47118.07	2.75
		BA	206365.14	315540.73	248597.91	36993.35	10.16
1	**38D**	GA	175777.01	318692.55	245159.36	48239.59	3.30
		TS	**146591.19**	375931.78	245933.05	70696.88	3.26
		DVBA	147437.82	291735.15	**240870.98**	43251.77	2.06
		PSO	**166811.41**	484921.82	313849.03	65005.95	5.55
		BA	202757.68	489685.77	329012.12	71927.23	7.44
2	**54D**	GA	182540.68	486771.12	306981.52	85964.83	5.22
		TS	210657.29	454798.29	319737.95	61750.83	4.01
		DVBA	203342.20	**412634.10**	**299536.05**	61679.63	5.74
		PSO	355932.28	728729.55	527034.60	85947.92	10.80
		BA	285515.59	**652412.35**	475560.82	87421.48	12.22
3	**72D**	GA	294449.33	775318.44	516123.15	113287.31	12.41
		TS	278259.26	702875.83	**468966.32**	100058.71	12.88
		DVBA	**272455.06**	749063.12	470188.66	110948.31	11.17
		PSO	485451.46	**931690.59**	727803.87	105982.11	15.78
		BA	519731.62	1175658.75	779329.67	129138.17	15.22
4	**96D**	GA	511018.33	1079571.97	722368.44	127095.75	16.41
		TS	563112.71	973503.78	737951.79	112591.61	15.88
		DVBA	**437008.44**	985441.12	**703656.08**	131965.85	15.17
		PSO	679465.79	**1105596.08**	883460.38	130587.06	25.54
		BA	638456.13	1241163.81	885037.02	146848.01	27.32
5	**120D**	GA	626694.36	1187243.80	863766.70	157404.16	28.67
		TS	635904.20	1163537.74	866045.89	128691.68	26.18
		DVBA	**506594.61**	1117955.20	**848871.39**	157256.42	26.17

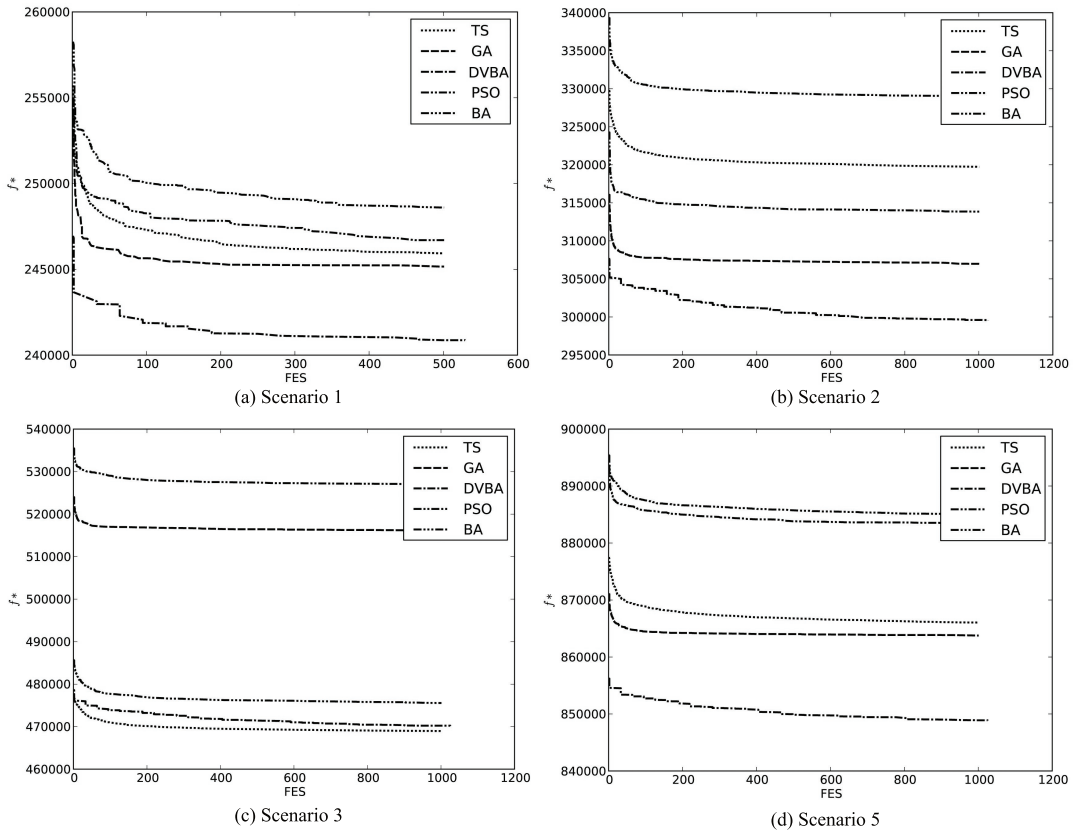


Figure 5.7: Convergence characteristics of PSO, BA, GA, TS, and DVBA on different scenarios of supply chain cost problem.

From the experimental results we can say that DVBA coped with this balance problem successfully. Although dimensions get very high in some scenarios of supply chain, the results of the DVBA were better than other algorithms. This comparison showed that DVBA is not only good at finding global optimum of some benchmark functions, it is also good at complicated real world problem.

CHAPTER 6

IMPROVEMENTS ON DYNAMIC VIRTUAL BATS ALGORITHM

Bat Algorithm is very good at exploitation; however, it is generally poor at exploration. On the other hand, Dynamic Virtual Bats Algorithm is very efficient in exploration and exploitation, but it still needs improvements, when it comes to high dimensional problems. In this section, a novel micro-bat algorithm (μBA) [97] is presented which possess the advantages of both algorithms. μBA employs a very small population compared to its classical version. It combines the swarming technique of bats in Bat Algorithm with the role based search in Dynamic Virtual Bats Algorithm. Our empirical results demonstrate that the proposed μBA achieves a good balance between exploration and exploitation. And it exhibits a better overall performance than the standard BA with larger and smaller populations on high dimensional problems.

6.1 Micro Bat Algorithm

μBA is developed using the ideas from the Bat Algorithm and Dynamic Virtual Bats Algorithm. In this combination, the weaknesses of the algorithms are avoided and the advantages are used. The weaknesses and the advantages of the BA and the DVBA can be summarized as follows.

In Bat Algorithm, the bats repeat three main steps as the iterations proceed.

1. Bats move towards the best found position.

2. Bats, with a probability of $rand() < r$, fly near to the best position.

3. Bats fly randomly either near to the best bat or any position in the search space.

Here, r is the sound impulse rate and it increases exponentially as the iterations proceeds (Eq 3.6). It is clear that, the possibility of flying near to the best position will increase rapidly for each bat after some iterations [42]. In another word, Bat Algorithm loses its exploration capability rapidly and increases its exploitation capability at the following iterations. That can cause Bat Algorithm to converge prematurely [111].

In DVBA, there are two roles which are exchanged between the bats according to their positions during the search. This dynamic role exchange gives DVBA higher diversity capability but slower convergence [92] [94]. The size of the bats' search scope has a major affect on DVBA's performance. The search scope size is limited by the wavelength which might not be long enough to detect better solutions near its surrounding space. Thus, it is very likely that the explorer bat will be trapped in local optima. Additionally, the exploiter bat's search scope can become very small during the exploitation process and it will move very slowly. Therefore it might not reach the global optima within the bounded computation time.

In the μBA, the position of the prey represents a possible solution to the optimization problems. To discover the prey, three bats are employed which are referred as explorer bat, exploiter bat, and scout bat. The explorer and the exploiter bats show the same characteristics as in DVBA. However, they do not exchange the roles during the search and the explorer bat helps the exploiter bat to speed up the exploitation. The scout bat was added to increase search diversity. The μBA combined the BA's fast convergence characteristic with DVBA's exploration and dynamic search capabilities. The behavior of the virtual bats and the outline of the μBA are given below.

6.1.1 The explorer bat

The explorer bat emits the sound pulses with low frequency and long wavelength so it can scan a large area (Fig. 3.2a). The explorer bat checks the solutions on its search scope and flies

to the best solution. Unless there is no better solution than its current position, the explorer bat will turn around randomly and keep scanning its nearby surrounding space until it finds a better solution. So it is clear that it can be trapped easily in local optima like in DVBA. To avoid this local optima trap, we give the explorer bat a chance to make a random fly in the vicinity of the exploiter bat (Eq 6.2). That will not only help the explorer bat to escape from the trap but also increase the speed of the exploiter bat through the global optima. The similar probabilistic approach ($rand() > ri$) from the Bat Algorithm is used to give this chance to the explorer bat. In μBA, r_i is switched by P and called the random flight probability. The probability of random flight P and the new position of the bat are calculated as follows:

$$P^{t+1} = P^t[1 - exp(-trial\sigma)], \tag{6.1}$$

$$x_i^{t+1} = x_{gbest} + \rho, \tag{6.2}$$

where σ constant. *trial* denotes the number of unsuccessful attempts to escape from the local optima trap. ρ is calculated as in Eq. 4.14. As the explorer bat spins around the local optima trap, the P will decrease exponentially and the probability of flying near to the exploiter bat ($rand() > P$) will increase. In Eq.6.1, if σ is chosen very small, the explorer bat will fly near to the exploiter bat too soon and will not able to explore its nearby surrounding space. Thus, σ should be chosen carefully. As the unsuccessful attempts (trial) increases, the effect of σ on P is shown in Fig.6.1.

6.1.2 The exploiter bat

The exploiter bat is used to increase the intensification of the search on the best found solution. It has very narrow search scope (Fig. 6.3) so it can make intense exploitation. If the explorer bat or the scout bat finds a better solution than the exploiter bat's current position, it flies to this solution and starts exploiting there.

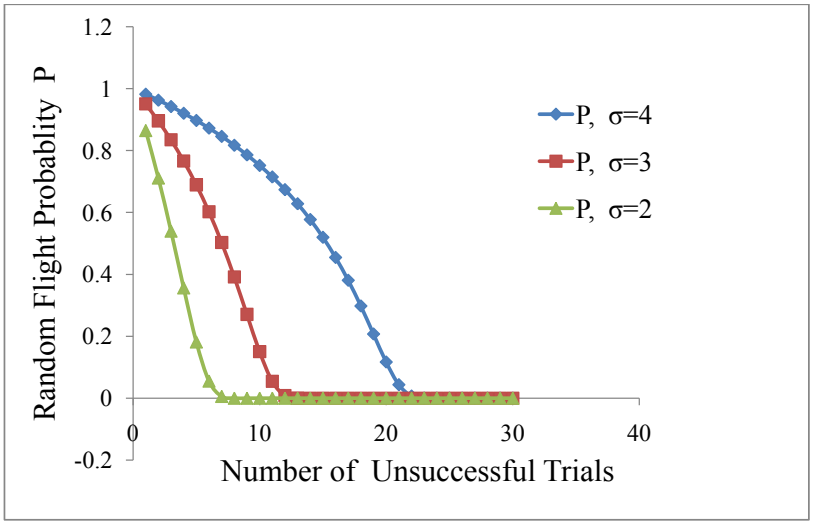


Figure 6.1: The effect of σ on P as the unsuccessful attempts increases.

6.1.3 The scout bat

The scout bat has a large search scope like the explorer bat (Fig. 6.2). However, unlike the explorer bat, the scout bat does not consider its current position to make the next move. It simply chooses the best position from its search scope and flies there. Same as in Simulated Annealing (SA) [14], it will even fly to worse solution than its current solution. The scout bat keeps flying all around the search space without having any local optima trap problem. That increases the diversification capability of the μBA.

In a robust search process, exploration and exploitation processes must be carried out together. In the μBA, while explorer and exploiter bat carry out the exploitation process in the search space, the scout bat control the exploration process with the explorer bat.

According to all these approximations and improvements μBA can be given as in Algorithm3.

Algorithm 3 μBA pseudo code where x_{gbest} is the global best position and d is the number of dimensions.

Objective function *f(x)*, $x = (x_1, ..., x_d)^T$
Initialize the bat population $x_i (i = 1, 2, 3)$ and velocity v_i
Initialize wavelength λ_i and frequency f_i
Initialize the number of the wave vectors and search points (r, k)
while ($t <$ Max number of iterations) **do**
  for each bat **do**
    Create a sound waves scope [92]
    Evaluate the solutions on the waves
    Choose the best solution on the waves, h_*
  if ($i = 1$) **then** //Scout Bat
    Move to h_*
    Maximize λ_i and minimize f_i
  end if
  if ($i = 2$) **then** //Explorer Bat
    if $f(h_*) < f(x_i)$ **then**
      Fly to h_*
      Maximize λ_i and minimize f_i
      trial=0
    else
      trial=trial+1
      Change the direction randomly
      Update P by using Eq. 6.1
    end if
    if ($rand() > P$) **then**
      Produce a new solution around the exploiter bat
      by Eq. 6.2
      $trial = 0$
      $P = 1$
    end if
  end if
  if ($i = 3$) **then** //Exploiter Bat
    Fly to x_{gbest}
    Minimize λ_i and maximize f_i
    Change the direction randomly
  end if
  Rank the bats and find the current best x_{gbest}
end while

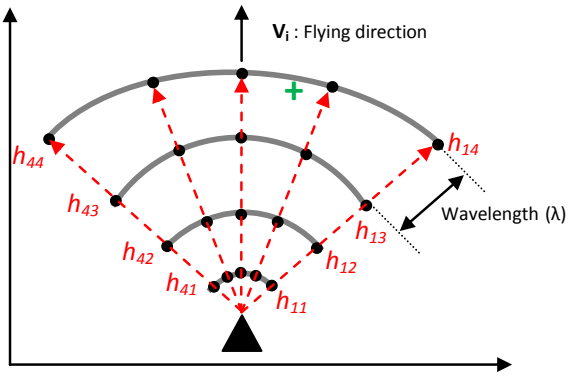


Figure 6.2: Exploration: Explorer bat is searching for prey with a wide search scope.

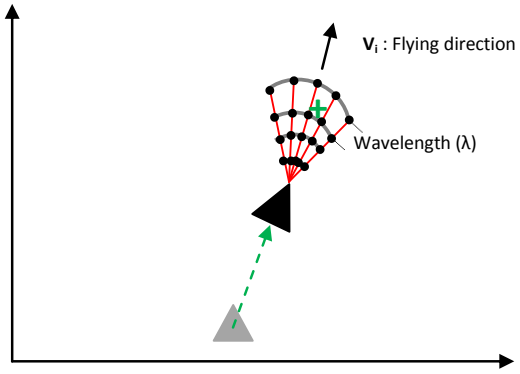


Figure 6.3: Exploitation: Exploiter bat is chasing prey with a narrow search space.

Table 6.1: Description of the benchmark functions. Here D: dimensionality of the functions, C: function characteristics with values - U: unimodal, M: Multimodal, S: Separable, N: Non-Separable.

No	Name	Formula	D	C	f_{min}	Search Space
f_1	Ackley	$f_1(x) = 20 + e - 20exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - exp(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i))$	10, 30, 50	MN	0	$(-32, 32)^d$
f_2	Bohachevsky	$f_2(x) = \sum_{i=1}^{d-1}[x_i^2 + 2x_{t+1}^2 - 0.3cos(3\pi x_i) - 0.4cos(4\pi x_{i+1}) + 0.7]$	10, 30, 50	MN	0	$(-15, 15)^d$
f_3	Griewangk	$f_3(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos(\frac{x_i}{\sqrt{i}}) + 1$	10, 30, 50	MN	0	$(-600, 600)^d$
f_4	Rastrigin	$f_4(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)]$	10, 30, 50	MS	0	$(-5.12, 5.12)^d$
f_5	Powell	$f_{16}(x) = \sum_{i=1}^{d-2}(x_{i-1} + 10x_i)^2 + 5(x_{i+1} - x_{i+2})^2 + (x_i - 2x_{i+1})^4 + 10(x_{i-1} - x_{i+2})^4$	10, 30, 50	UN	0	$(-4, 5)^d$
f_6	Rosenbrock	$f_5(x) = \sum_{i=1}^{d-1}[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	10, 30, 50	MN	0	$(-15, 15)^d$
f_7	Sphere	$f_{13}(x) = \sum_{i=1}^{d} x_i^2$	10, 30, 50	US	0	$(-5.12, 5.12)^d$
f_8	Shifted Rastrigin	$f_8(x) = f_4(z),\ z = x - o,\ o = [o_1, o_2, ..., o_d]$: shifted global optimum.	10, 30, 50	US	0	$(-5.12, 5.12)^d$
f_9	Shifted Rotated Ackley	$f_9(x) = f_1(z),\ z = M(x - o),\ o = [o_1, o_2, ..., o_d]$: shifted global optimum	10, 30, 50	UN	0	$(-32, 32)^d$
f_{10}	Shifted Rotated Griewangs	$f_{10}(x) = f_3(z),\ z = M(x - o),\ o = [o_1, o_2, ..., o_d]$: shifted global optimum	10, 30, 50	US	0	$(-600, 600)^d$

In the next section, experimental results on high-dimensional instances of widely used optimization problems are reported.

6.2 Numerical Experiments and Results

6.2.1 Parameter settings for the algorithms

In order to demonstrate the effectiveness of the μBA, a suite of 10 well-known numerical functions were tested with DVBA, BA with 30 bats, and BA with 3 bats. Each test function was considered three different dimensions, namely, $d = 10, 30$, and 50. Maximum number of function evaluation (FEs) is set to 100.000 for $10 - D$ problems, 300.000 for $30 - D$ problems, and 500.000 for $50 - D$ problems. Algorithms were tested with 30 independent runs for each test functions in order to compile comprehensive data. All the algorithms are developed in the Python environment and run on a PC with a 3.20 GHz CPU and 6.00 GB of RAM.

The other specific parameters of algorithms are given below:

BA Settings

Parameters are set as follows: $\alpha = \gamma = 0.5$, frequency is in the range $[0, 2]$, the rate of pulse emission $r \in [0, 1]$ and, the loudness $A_0 = 0.5$ [106].

DVBA Settings

Maximum step size divisor is set to $\beta = 250$ since the search space exponentially increased in our tests. The number of search points and the wave vectors are set to $r = 6$ and $k = 5$, respectively. The range of the wavelength and the frequency are set as follows: $[\lambda_{min}, \lambda_{max}] = [\rho, 10\rho]$ and $[f_{min}, f_{max}] = [\rho, 10\rho]$, where ρ is calculated in Eq. 4.14. Population size is 2.

micro-BA Settings

Same parameters are used from DVBA to create the search scope of the bats. P is started from 1 and $\sigma = 3$.

6.2.2 Benchmark Functions

To evaluate the performance of the algorithms, a set of 10 standard benchmark functions is used. The benchmark set include unimodal, multimodal, separable, non-separable, shifted, and rotated optimization functions. In shifted and rotated test functions, the global optimum is shifted to a random position and the functions are rotated. By using shifted and rotated functions, we would be able to test the algorithms on more challenging, real world like problems [57]. Specifically, functions $f_1 - f_4$ are multimodal functions, $f_4 - f_7$ are unimodal functions, and $f_8 - f_{10}$ are shifted and rotated functions. We rotated the functions $f(z) = f(Mx)$, where $f(z)$ is the new function and M is an orthogonal rotation matrix. The global optimum is shifted to a random position by $f(z) = f(x - o_{new} - o_{old})$, where o_{old} is the old global optimum and o_{new} is the new global optimum which is not lying at the center of the search range [75]. The description of the benchmark functions are shown in Table 6.1.

6.2.3 Experimental results and discussion

In order to test the efficiency of the proposed algorithm, our experiment's results were compared with the standard BA with 3 bats, the standard BA with 30 bats, and the standard DVBA. The test results are shown in Table 2 and 3 in terms of the best fitness values (BFV), the worst fitness values (WFV), the mean and the standard deviation (STDEV) of the results found over the 30 independent runs by each algorithm. The best results are marked in bold. Furthermore, we used t-tests [14] to compare the means of the results produced by the μBA and the other algorithms at the 0.05 level of significance. In the last column of Table 2 and 3 we report the statistical significance level of the results. There $'-'$ indicates that μBA is significantly more successful than selected one at a 0.05 level of significance by two-tailed test, $' \approx'$ means the

Table 6.2: Performance comparison of DVBA, BA with 3 bats, BA with 30 bats, and μBA over 10 test functions of 10, 30, and 50 dimensions.

	Function	Dim	Algorithms	BFV	WFV	Mean	STDEV	t-test
f_1	Ackley	10	DVBA	0.0216	4.5465	2.6865	0.9056	–
			BA-3bats	2.3247	4.6683	3.5937	0.6870	–
			BA-30bats	2.0193	4.0331	2.8221	**0.4931**	–
			μBA	**0.0146**	**3.0281**	**1.7313**	1.0036	
		30	DVBA	**3.0604**	20.0798	16.6419	6.7907	–
			BA-3bats	19.9420	19.9667	19.9568	0.0113	–
			BA-30bats	19.9434	19.9566	19.9497	**0.0042**	–
			μBA	3.2660	**5.0264**	**3.8358**	0.5119	
		50	DVBA	20.0848	20.2278	20.1788	0.0514	–
			BA-3bats	19.9160	19.9660	19.9483	**0.0171**	–
			BA-30bats	3.5297	19.9636	18.3037	4.9246	–
			μBA	**2.8373**	**5.4899**	**4.4476**	0.7227	
f_2	Bohachevsky	10	DVBA	1.5428	4.4134	2.9063	0.7375	–
			BA-3bats	1.9253	2.9166	**2.2136**	**0.2793**	$\approx$
			BA-30bats	1.7723	**2.8279**	2.3783	0.3067	$\approx$
			μBA	**0.2317**	4.4773	2.3240	1.1738	
		30	DVBA	14.7338	26.1569	19.7721	3.9088	–
			BA-3bats	17.4313	20.3219	18.4629	**1.0059**	$\approx$
			BA-30bats	17.5538	22.1077	19.3455	1.6834	–
			μBA	**13.3856**	**20.0175**	**17.9844**	2.4474	
		50	DVBA	37.6436	44.4501	42.1866	2.5494	–
			BA-3bats	38.0999	44.1018	41.5410	**2.1463**	–
			BA-30bats	39.2839	44.8488	42.8278	1.9773	–
			μBA	**24.4439**	**38.2343**	**31.9908**	4.8434	
f_3	Griewangk	10	DVBA	1.3576	2.4926	2.1272	0.3303	–
			BA-3bats	0.5478	0.9346	**0.7585**	**0.0941**	$\approx$
			BA-30bats	0.5217	**0.8826**	0.7510	0.1073	$\approx$
			μBA	**0.7075**	0.8509	0.7980	0.0434	
		30	DVBA	10.4302	13.7109	12.9437	1.2662	–
			BA-3bats	1.3904	1.4195	1.4067	**0.0095**	–
			BA-30bats	1.3501	1.4370	1.4050	0.0296	–
			μBA	**1.0804**	**1.1049**	**1.0928**	0.0104	
		50	DVBA	33.3351	38.2587	35.6369	1.7511	–
			BA-3bats	2.1307	2.2624	2.2165	0.0471	–
			BA-30bats	1.7346	2.1055	1.9677	0.1245	–
			μBA	**1.2277**	**1.2935**	**1.2605**	**0.0215**	

Table 6.3: Performance comparison of DVBA, BA with 3 bats, BA with 30 bats, and μBA on Rastrigin, Powell, and Rosenbrock with 10, 30, and 50 dimensions.

	Function	Dim	Algorithms	BFV	WFV	Mean	STDEV	t-test
f_4	Rastrigin	10	DVBA	17.2326	**41.9676**	30.0917	**8.6982**	$\approx$
			BA-3bats	56.1000	110.5451	85.3065	19.8476	–
			BA-30bats	18.3934	69.8718	50.2790	18.2945	–
			μBA	**15.2605**	58.9553	**27.1348**	16.3371	
		30	DVBA	**141.0519**	243.0097	**195.9515**	33.3322	–
			BA-3bats	176.5749	293.3492	235.6283	37.8489	–
			BA-30bats	183.9583	245.1114	221.4598	20.2705	–
			μBA	183.6387	**210.9989**	198.0123	**9.9362**	
		50	DVBA	394.2856	484.7279	444.4586	29.2374	–
			BA-3bats	370.7712	497.3180	440.2515	57.1338	–
			BA-30bats	**349.3720**	483.0212	407.0945	53.4919	–
			μBA	376.2547	**431.6168**	**406.1995**	**21.9270**	
f_5	Powell	10	DVBA	0.0953	0.1916	0.1400	0.0253	–
			BA-3bats	0.0660	0.1715	0.1153	0.0283	–
			BA-30bats	0.0682	0.1743	0.1001	0.0301	–
			μBA	**0.0075**	**0.0162**	**0.0114**	**0.0027**	
		30	DVBA	0.9749	1.1569	1.0879	0.0629	–
			BA-3bats	1.4853	2.1225	1.8663	0.2397	–
			BA-30bats	1.6184	2.0971	1.9913	0.1867	–
			μBA	**0.0908**	**0.1183**	**0.1032**	**0.0107**	
		50	DVBA	2.6277	3.6723	3.1850	0.4118	–
			BA-3bats	4.9632	6.3985	5.6586	0.4875	–
			BA-30bats	4.8066	6.3320	5.5817	0.4941	–
			μBA	**0.2807**	**0.3442**	**0.3176**	**0.0239**	
f_6	Rosenbrock	10	DVBA	11.3205	166.3953	29.0556	45.8787	–
			BA-3bats	10.6727	196.4292	46.6597	68.0033	–
			BA-30bats	10.5964	17.4026	13.4332	2.3198	–
			μBA	**4.1786**	**10.2598**	**8.9103**	**1.6942**	
		30	DVBA	85.7328	1072.1944	299.3281	386.9613	–
			BA-3bats	96.5783	143.2552	126.3990	**16.9769**	–
			BA-30bats	97.8295	427.0891	238.8529	139.9443	–
			μBA	**31.3629**	**87.8815**	**43.5059**	22.2007	
		50	DVBA	240.6517	1269.5626	579.9789	327.6391	–
			BA-3bats	298.4399	861.5234	572.4530	176.6554	–
			BA-30bats	323.3869	972.9987	450.1929	186.3294	–
			μBA	**61.4320**	**314.5447**	**165.5107**	**79.8678**	

Table 6.4: Performance comparison of DVBA, BA with 3 bats, BA with 30 bats, and μBA over Sphere, Shifted Rastrigin, and Shifted Rotated Ackley function with 10, 30, and 50 dimensions.

	Function	Dim	Algorithms	BFV	WFV	Mean	STDEV	t-test
f_7	Sphere	10	DVBA	0.2612	**0.5413**	0.3739	0.0911	−
			BA-3bats	1.3693	3.6073	2.8133	0.6197	−
			BA-30bats	1.3021	3.0094	2.3567	0.5229	−
			μBA	**5.65e-07**	0.6860	**0.1326**	**0.2366**	
		30	DVBA	0.0553	0.0759	0.0643	0.0077	−
			BA-3bats	0.1010	0.1320	0.1133	0.0102	−
			BA-30bats	0.1025	0.1232	0.1116	0.0085	−
			μBA	**0.0111**	**0.0135**	**0.0121**	**0.0009**	
		50	DVBA	0.1497	0.2200	0.1832	0.0230	−
			BA-3bats	0.2781	0.3822	0.3373	0.0281	−
			BA-30bats	0.3020	0.3799	0.3408	0.0216	−
			μBA	**0.0132**	**0.0206**	**0.0175**	**0.0019**	
f_8	Shifted Rastrigin	10	DVBA	28.9949	127.0091	72.8769	22.4583	−
			BA-3bats	46.2250	188.3732	103.7364	40.2252	−
			BA-30bats	25.0820	92.8895	57.0023	18.4316	−
			μBA	**11.0428**	**76.7472**	**39.8817**	**17.9924**	
		30	DVBA	211.0952	498.3511	368.8925	83.9036	−
			BA-3bats	297.7370	653.7204	460.0270	90.4837	−
			BA-30bats	250.8744	469.1721	337.1287	65.2766	−
			μBA	**148.7219**	**314.5511**	**232.5197**	**55.4156**	
		50	DVBA	606.4386	821.6168	708.6816	**62.4324**	−
			BA-3bats	692.8852	1009.7283	847.5367	110.5501	−
			BA-30bats	644.1346	837.0152	720.2708	68.0795	−
			μBA	**321.5644**	**610.9722**	**462.8984**	106.1891	
f_9	Shifted Rotated Ackley	10	DVBA	1.7521	20.3361	7.4764	8.2766	+
			BA-3bats	2.2921	20.4352	19.3113	**3.2061**	−
			BA-30bats	1.9483	20.4179	14.8852	8.3534	−
			μBA	**0.4599**	**20.2919**	**5.3013**	7.5357	
		30	DVBA	20.8117	21.0153	20.9279	0.0548	≈
			BA-3bats	20.8161	21.0337	20.9658	0.0595	≈
			BA-30bats	20.8892	21.0349	20.9639	**0.0421**	≈
			μBA	**2.8299**	**20.7155**	**17.0460**	7.1056	
		50	DVBA	21.0451	21.1607	21.1241	0.0308	−
			BA-3bats	21.1297	21.1911	21.1597	**0.0195**	−
			BA-30bats	21.0353	21.1986	21.1363	0.0448	−
			μBA	**20.7314**	**20.8755**	**20.7972**	0.0491	

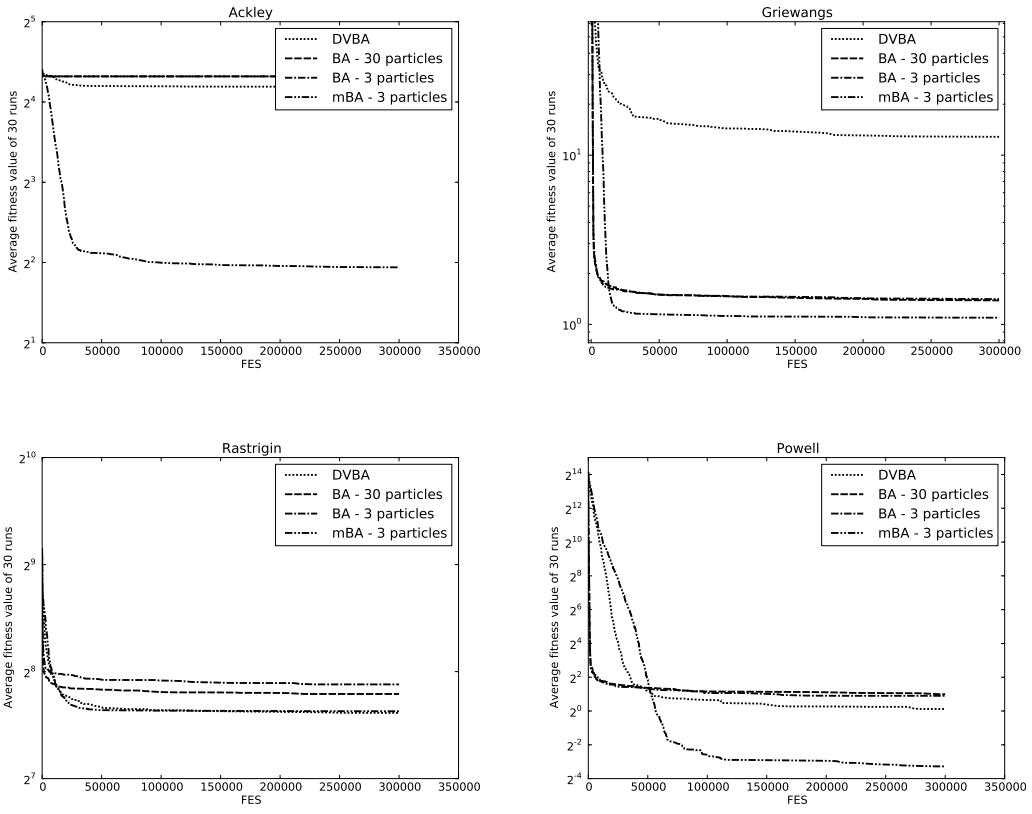


Figure 6.4: Convergence characteristics of μBA, BA, and DVBA for the 30-dimensional Ackley, Griewangs, Rastrigin, and Powell functions.

Table 6.5: Performance comparison of DVBA, BA with 3 bats, BA with 30 bats, and μBA Shifted Rotated Griewangs with 10, 30, and 50 dimensions.

	Function	Dim	Algorithms	BFV	WFV	Mean	STDEV	Significant
f_{10}	Shifted Rotated Griewangs	10	DVBA	2.2979	3.7364	3.0175	0.4656	–
			BA-3bats	0.8388	1.0362	0.9506	**0.0630**	–
			BA-30bats	0.8025	1.0262	0.9003	0.0850	–
			μBA	**0.4069**	**0.9148**	**0.7429**	0.1519	
		30	DVBA	7.2322	8.7454	7.3364	1.9442	–
			BA-3bats	1.6303	1.9125	1.7939	0.0934	–
			BA-30bats	1.6556	1.8832	1.7739	0.0766	–
			μBA	**1.1605**	**1.2358**	**1.2008**	**0.0221**	
		50	DVBA	10.8454	14.9527	11.6678	2.5123	–
			BA-3bats	3.2262	3.7766	3.4695	**0.1737**	–
			BA-30bats	3.0935	3.9139	3.4128	0.2344	–
			μBA	**1.4815**	**1.6367**	**1.5713**	0.0453	

difference of means is not statistically significant and, $'+'$ stands for Not Applicable, covering cases for which the two algorithms achieve the same accuracy results.

Fig.6.4 and 6.5 illustrates the convergence characteristics in terms of the best fitness value of the median run of each algorithm for the test functions with $D = 30$. The convergence graphs of the $10-D$ and $50-D$ problems are similar to their $30-D$ counterparts, so they are omitted here to save space.

From Table 6.3, 6.4, and 6.5 it can be said that μBA gave the best results for multimodal test functions. It is known that the complexity of the problems increases as the dimensionality of the search increases and local traps become harder to escape for the algorithms. However, the μBA performed successfully for the multimodal functions f_1, f_2, and f_3 for all dimensions. For the functions f_4, the algorithms did not show a significant success but μBA has demonstrated a better ability of global searching. For the unimodal functions, the algorithms got the best results but they performed poorly on f_8 (Rosenbrock). Rosenbrock function is grouped as a unimodal function but it may have multiple minima when the dimension increases and converging to the minimum is difficult [82]. Furthermore, the t-test values show that the performance of the μBA is significantly more efficient than other compared algorithms in terms of the mean. Similar

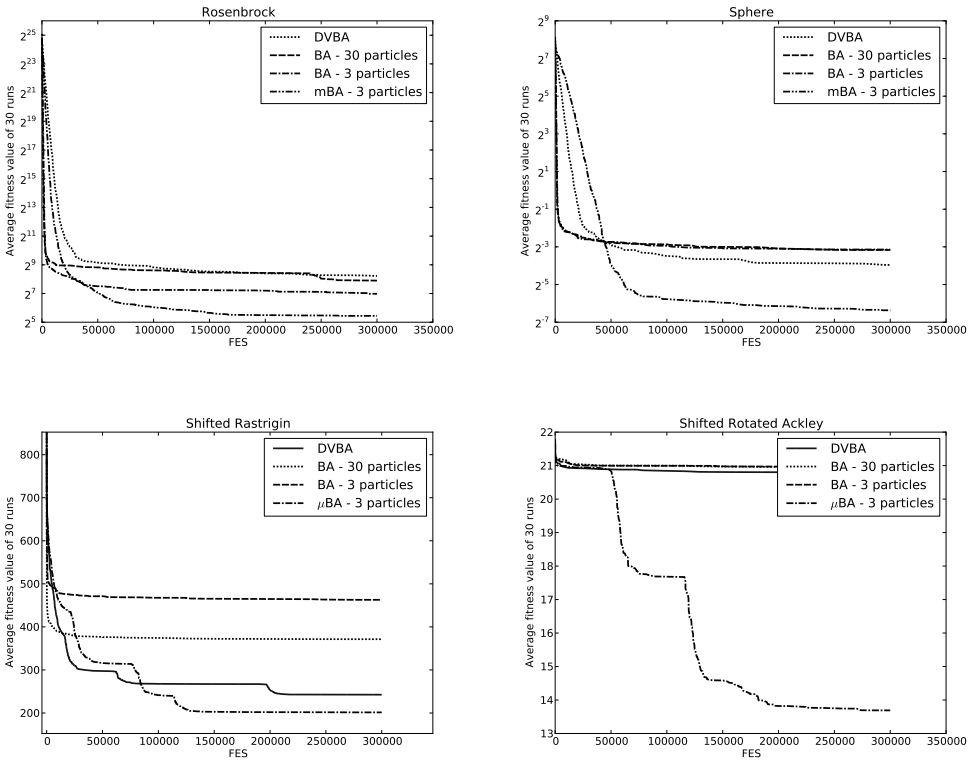


Figure 6.5: Convergence characteristics of μBA, BA, and DVBA for the 30-dimensional Rosenbrock, Sphere, Shifted Rastrigin, and Shifted-Rotated Ackley functions.

observations can be made for the shifted and rotated functions. Overall, μBA obtains a smaller mean value than the other algorithms for all problems.

Additionally, a close inspection of tables reveals that, BA does not sensitive to the population size. BA with 3 bats and BA with 30 bats did not show significant difference on most of the test functions.

The convergence map of algorithms in Fig.6.4 and 6.5 shows that the μBA always converges faster than other algorithms on seven problems (f_1, f_5, f_6, f_7, f_8, f_9, and f_{10}). In Fig.6.4 and 6.5, it can be seen that, while other algorithms suffer from premature convergence problem on the function f_1, f_5, f_7, f_8, and f_9, the μBA escaped from the local optima traps successfully. For the functions f_1 and f_9, μBA outperformed the other algorithms significantly. f_1 and f_9 is characterized by a nearly flat outer region, and a large hole at the center. The function poses a risk for the algorithms to be trapped in one of its many local minima and difficulty to reach the global optima in predefined time. The explorer bat helped the exploiter bat to reach the global optima faster in these functions.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

The conclusions from this thesis are collected in this chapter. Moreover, in the last part of this chapter, we present some thoughts on potential future work that could extend on this thesis.

7.1 Conclusions

In this thesis, a novel nature inspired metaheuristic optimization algorithm, Dynamic Virtual Bats Algorithm (DVBA), is proposed. The objective proposed for this thesis were the following:

1. Make an in-depth study of echolocation and bat algorithm.
2. Develop a new algorithm by imitating the bat's hunting strategies that can manage the exploration and exploitation conflict in a effective and efficient way.
3. Validate the performance of the proposed algorithm DVBA by comparing with classic and state-of-the-art optimization algorithms.
4. Analyze the performance of dynamic virtual bats algorithm on a real world problem.
5. Study on possible improvements for dynamic virtual bats algorithm by using the advantages of the bat algorithm and analyze it.

With regards to objective 1, bat's hunting strategies and bat algorithm are investigated in Chapter 3. We have seen that bats are amazing creatures who can detect objects in pitch

black dark caves by using echolocation. They have ability to change frequency and wavelength of the emitted sound waves. During the hunt, they can increase the rate of pulse emissions when they near their prey. They also emit sounds very loudly during exploration and become very quite near their prey. These are the main strategies which were the inspiration of the bat algorithm and proposed dynamic virtual bats algorithm. Although all these abilities of the real bats are listed in [106] for bat algorithm, a few of them are simulated in the algorithm. Bat algorithm is like a combination of PSO and harmony search. It can be concluded that BA lacks imitation of real bats. We also realized that BA suffers from the premature convergence problem and it needs improvements in exploration. By modifying random walk size and the pulse rate parameters in BA, it is possible to overcome this problem.

In Chapter 4, we have presented our new bat inspired algorithm, DVBA (Objective 2). It is conceptually very different from BA. DVBA is a new simulation of bat's hunting strategies. One novelty of DVBA was the population size. DVBA needs just two bats to overcome the exploration and exploitation trade off problem. They are called the explorer and the exploiter bats. While the explorer bat scans large area roughly, the exploiter bat increase the intensification of the search around the best found solution. That helps bats to avoid from local optima traps. Another novelty was how DVBA explores the search space. The virtual bats in the algorithm behave the same as the real bats in nature when hunting.

The overall efficiency and performance of a metaheuristic algorithm is up to its fine balance between diversification (exploration) and intensification (exploitation). To analyze how DVBA is coping with this problem, in Chapter 5, we have compared the DVBA with 4 standard optimization algorithms, 4 modified BA algorithms, and 5 state-of-the-art algorithms competed in special session at CEC 2014 over 6 classic and 30 CEC 2014 single objective optimization test functions. From the experimental results we can say that DVBA coped with this balance problem successfully. Objective 3 is accomplished in Chapter 5.

To fulfill the objective 4, DVBA has been applied to minimize the supply chain cost problem. Since finding the minimal cost associated with a globally distributed supply chain is an NP-hard problem, metaheuristics algorithms are considered a good approach to give an optimized result.

Therefore, the performance of DVBA was tested on supply chain cost problem with other well-known algorithms; Particle Swarm Optimization (PSO), Bat Algorithm (BA), Genetic Algorithm (GA), and Tabu Search (TS). The result of the case study showed that the DVBA is much superior to other algorithms in terms of accuracy and efficiency. Although BA, PSO, and GA start the optimization with 30 particles, they could not get better cost then DVBA in general. Because of its complexity, after a few assessments time algorithms reach their global optimum and then they trap there. However DVBA manages the exploration better than other algorithms and converges in a better global optimum.

While we have been studying the effectiveness of DVBA, we realized that the exploitation ability of DVBA can be improved by using some features of bat algorithm. Therefore, in Chapter 6, we proposed an improved version of DVBA named micro-bat algorithm (μBA). In μBA, the proposed new search mechanism combines the BA's fast convergence characteristic with DVBA's dynamic search capability to increase the search diversity while pursuing a balance between exploration and exploitation.

To prove the effectiveness and robustness of the proposed algorithms, the μBA was compared with DVBA and BA on variety of optimization problems with different complexities. The results demonstrated that the solution values achieved by μBA are several orders of magnitude better than BA and comparable to DVBA in many cases. Also, μBA appears to be less effected when dimension of the problem increases significantly. Therefore, it can be said that μBA achieves a good balance between exploration and exploitation and has the best universality on different type of problems.

Finally, We can conclude that the proposed DVBA could evolve suitable strategies and parameter values as evolution progress by using just two bats. Exchanging the roles of the bats dynamically helped bats to escape from local optima traps. In general, DVBA outperforms or is comparable to other algorithms on most of the test functions. It is also good at complicated real world problems such as supply chain cost. We can see that it is a very promising algorithm and suitable for improvements.

7.2 Future Work

There are several directions for future work. There might be a refinement on DVBA's exploration ability. We have seen that DVBA does not perform well on most of the high-dimensional rotated functions or composition functions.

Since DVBA has just two bats, DVBA lags behind other algorithms in terms of convergence speed at the beginning of the FEs, although DVBA outperforms significantly after some learning period. Especially for high-dimension problems, escaping from local optima traps becomes a challenge. In DVBA, the explorer bat's duty is to cope with this problem, but if it stuck in a large optima trap, its wavelength might not be long enough to discover a better position and escape from the trap.

As future work, the explorer bat's history can be stored, like in Tabu Search (TS). After some unsuccessful attempts to escape from these local optima traps the explorer bat might restart its search again, but use previous failed experience to not waste function evaluations. Of course this will slow down the computation time, but convergence speed might increase. And it will help the explorer bat to explore the search space more efficiently.

Furthermore, as bats use time difference between their two ears to generate a 3D blueprint of their environment, they can distinguish the shape, size, and texture of a tiny prey, in which direction the prey is heading, and even the speed of the prey. Therefore, a further natural extension to the current DVBA would be to use Doppler effect [10, 81], which may lead to new algorithms to solve dynamic optimization problems.

Also, parallel programming can be used as a future work to increase the speed of the algorithm. In that case, bats will communicate simultaneously and increasing the number of bats can be considered.

Appendix A

VERIFICATION OF OUR FRAMEWORK RESULTS

Table A.1: Comparison of PSO and BA in our framework with the results in [100]

	Results from [15]				Our Program Results			
	PSO		BA		PSO		BA	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Circles Function	2.581	2.233	0.3862	0.1583	1.7371	0.7297	0.3019	0.1202
Rosenbrock function	1.8E+5	1.2E+6	2.8E+5	2.9E+5	1.9E+5	1.7E+6	2.9E+5	4.0E+4
Griewank's function	0.1538	0.0988	2.3832	0.4154	5.0624	4.518	2.2881	0.5754
Ackley	1.8191	0.4036	8.1288	1.8181	3.8685	0.3191	9.9282	0.0106
Rastrigin	185.0561	33.5121	99.4527	21.1198	212.3899	42.8747	85.9725	20.3178

REFERENCES

[1] Aarts, E., and Korst, J. Simulated annealing and boltzmann machines: a stochastic approach to combinatorial optimization and neural computing.

[2] Abd-Elazim, S., and Ali, E. Load frequency controller design via bat algorithm for nonlinear interconnected power system. *International Journal of Electrical Power & Energy Systems 77* (2016), 166–177.

[3] Adarsh, B., Raghunathan, T., Jayabarathi, T., and Yang, X.-S. Economic dispatch using chaotic bat algorithm. *Energy 96* (2016), 666–675.

[4] Airas, M. Echolocation in bats. In *Proceedings of spatial sound perception and reproduction. The postgrad seminar course of HUT Acoustics Laboratory* (2003), pp. 1–25.

[5] Arisha, A., and Abo-Hamad, W. Optimisation methods in supply chain applications: a review.

[6] Baziar, A., Kavoosi-Fard, A., and Zare, J. A novel self adaptive modification approach based on bat algorithm for optimal management of renewable mg. *Journal of Intelligent Learning Systems and Applications 5* (2013), 11.

[7] Beamon, B. M. Supply chain design and analysis:: Models and methods. *International journal of production economics 55*, 3 (1998), 281–294.

[8] Bujok, P., Tvrdik, J., and Polakova, R. Differential evolution with rotation-invariant mutation and competing-strategies adaptation. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 2253–2258.

[9] Chandrasekar, C., et al. An optimized approach of modified bat algorithm to record deduplication. *International Journal of Computer Applications 62*, 1 (2013).

[10] Chowning, J. M. The simulation of moving sound sources. *Journal of the Audio Engineering Society 19*, 1 (1971), 2–6.

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

[12] Cuevas, E., Cortés, M. A. D., and Navarro, D. A. O. Reduction of function evaluations by using an evolutionary computation algorithm. In *Advances of Evolutionary Computation: Methods and Operators*. Springer, 2016, pp. 121–152.

[13] Cuevas, E., Cortés, M. A. D., and Navarro, D. A. O. A states of matter algorithm for global optimization. In *Advances of Evolutionary Computation: Methods and Operators*. Springer, 2016, pp. 35–54.

[14] Das, S., Abraham, A., Chakraborty, U. K., and Konar, A. Differential evolution using a neighborhood-based mutation operator. *Evolutionary Computation, IEEE Transactions on 13*, 3 (2009), 526–553.

[15] Deb, K., and Agrawal, R. B. Simulated binary crossover for continuous search space. *Complex Systems 9*, 3 (1994), 1–15.

[16] Dixon, L. The choice of step length, a crucial factor in the performance of variable metric algorithms. *Numerical methods for non-linear optimization* (1972), 149–170.

[17] Dorigo, M. St utzle t. ant colony optimization, 2004.

[18] Dorigo, M., Birattari, M., and Stützle, T. Ant colony optimization. *Computational Intelligence Magazine, IEEE 1*, 4 (2006), 28–39.

[19] Dorigo, M., and Caro, G. The ant colony optimization meta-heuristic. in corne, d., dorigo, m. & glover, f.(eds.) new ideas in optimization, 1999.

[20] Dorigo, M., Maniezzo, V., and Colorni, A. The ant system: Optimization by a colony of cooperating agents. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B 26*, 1 (1996), 29–41.

[21] Dourado Maia, R., Nunes de Castro, L., and Matos Caminhas, W. Real-parameter optimization with optbees. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 2649–2655.

[22] Fang, W., Sun, J., Chen, H., and Wu, X. A decentralized quantum-inspired particle swarm optimization algorithm with cellular structured population. *Information Sciences 330* (2016), 19–48.

[23] Fiedler, J. Prey catching with and without echolocation in the indian false vampire (megaderma lyra). *Behavioral Ecology and Sociobiology 6*, 2 (1979), 155–160.

[24] Fister, I., Fong, S., Brest, J., and Fister, I. A novel hybrid self-adaptive bat algorithm. *The Scientific World Journal 2014* (2014).

[25] Fister, I., Rauter, S., Yang, X.-S., and Ljubič, K. Planning the sports training sessions with the bat algorithm. *Neurocomputing 149* (2015), 993–1002.

[26] Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research 13* (jul 2012), 2171–2175.

[27] Gao, H., Kwong, S., Yang, J., and Cao, J. Particle swarm optimization based on intermediate disturbance strategy algorithm and its application in multi-threshold image segmentation. *Information Sciences 250* (2013), 82–112.

[28] Geem, Z. W., Kim, J. H., and Loganathan, G. A new heuristic optimization algorithm: harmony search. *Simulation 76*, 2 (2001), 60–68.

[29] Glover, F. Future paths for integer programming and links to artificial intelligence. *Computers & operations research 13*, 5 (1986), 533–549.

[30] Glover, F. Tabu search-part i. *ORSA Journal on computing 1*, 3 (1989), 190–206.

[31] Goldberg, D. Genetic algorithms in search, optimization, and machine learning, addison-wesley, reading, ma, 1989. *NN Schraudolph and J.. 3*, 1.

[32] Goldberg, D. E. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex systems 5*, 2 (1991), 139–167.

[33] Griffin, D. R., Webster, F. A., and Michael, C. R. The echolocation of flying insects by bats. *Animal Behaviour 8*, 3 (1960), 141–154.

[34] Handfield, R. B., and Nichols, E. L. *Introduction to supply chain management*, vol. 1. prentice Hall Upper Saddle River, NJ, 1999.

[35] Holland, J. Adaptation in natural and artificial systems.

[36] Jaddi, N. S., Abdullah, S., and Hamdan, A. R. Multi-population cooperative bat algorithm-based optimization of artificial neural network model. *Information Sciences 294* (2015), 628–644.

[37] Jakobsen, L., Brinkløv, S., and Surlykke, A. Intensity and directionality of bat echolocation signals. *How nature shaped echolocation in animals* (2014), 72.

[38] Jakobsen, L., Ratcliffe, J. M., and Surlykke, A. Convergent acoustic field of view in echolocating bats. *Nature 493*, 7430 (2013), 93–96.

[39] Jakobsen, L., and Surlykke, A. Vespertilionid bats control the width of their biosonar sound beam dynamically during prey pursuit. *Proceedings of the National Academy of Sciences 107*, 31 (2010), 13930–13935.

[40] Jamil, M., and Yang, X.-S. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation 4*, 2 (2013), 150–194.

[41] Jordehi, A. R. A chaotic artificial immune system optimisation algorithm for solving global continuous optimisation problems. *Neural Computing and Applications 26*, 4 (2015), 827–833.

[42] Kabir, M. W. U., Sakib, N., Chowdhury, S. M. R., and Alam, M. S. A novel adaptive bat algorithm to control explorations and exploitations for continuous optimization problems. *International Journal of Computer Applications 94*, 13 (2014).

[43] Karaboga, D. An idea based on honey bee swarm for numerical optimization. Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[44] Karaboga, D. Artificial bee colony algorithm. *scholarpedia 5*, 3 (2010), 6915.

[45] Karaboga, D., and Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization 39*, 3 (2007), 459–471.

[46] Kaveh, A., and Zakian, P. Enhanced bat algorithm for optimal design of skeletal structures. *Asian J Civial Eng 15*, 2 (2014), 179–212.

[47] Kennedy, J., and Eberhart, R. C. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks* (Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995), vol. 4, pp. 1942–1948.

[48] Kephart, J. O., et al. A biologically inspired immune system for computers. In *Artificial Life IV: proceedings of the fourth international workshop on the synthesis and simulation of living systems* (1994), pp. 130–139.

[49] Khooban, M. H., and Niknam, T. A new intelligent online fuzzy tuning approach for multi-area load frequency control: Self adaptive modified bat algorithm. *International Journal of Electrical Power & Energy Systems 71* (2015), 254–261.

[50] Kumar, S. K., Tiwari, M., and Babiceanu, R. F. Minimisation of supply chain cost with embedded risk using computational intelligence approaches. *International Journal of Production Research 48*, 13 (2010), 3717–3739.

[51] Kwiatkowska, M., Mereacre, A., Paoletti, N., and Patanè, A. Synthesising robust and optimal parameters for cardiac pacemakers using symbolic and evolutionary computation techniques. In *Hybrid Systems Biology*. Springer, 2015, pp. 119–140.

[52] Lewis, R., Paechter, B., and Rossi-Doria, O. *Metaheuristics for university course timetabling*. Springer, 2007.

[53] Li, G., Niu, P., and Xiao, X. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. *Applied soft computing 12*, 1 (2012), 320–332.

[54] Li, J.-q., and Pan, Q.-k. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences 316* (2015), 487–502.

[55] Li, X., Tang, K., Omidvar, M. N., Yang, Z., Qin, K., and China, H. Benchmark functions for the cec 2013 special session and competition on large-scale global optimization. *gene 7* (2013), 33.

[56] Liang, J., Qu, B., and Suganthan, P. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. *Computational Intelligence Laboratory* (2013).

[57] Liang, J., Qu, B., Suganthan, P., and Chen, Q. Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore* (2014).

[58] Liang, J., Suganthan, P., and Deb, K. Novel composition test functions for numerical global optimization. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE* (2005), IEEE, pp. 68–75.

[59] Lin, J.-H., Chou, C.-W., Yang, C.-H., Tsai, H.-L., et al. A chaotic levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems. *Computer and Information Technology 2*, 2 (2012), 56–63.

[60] Luke, S. Essentials of metaheuristics, 2nd edn. lulu (2013), 1995.

[61] Luke, S. *Essentials of Metaheuristics*. Lulu, 2009. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[62] Mavrovouniotis, M., and Yang, S. Ant algorithms with immigrants schemes for the dynamic vehicle routing problem. *Information Sciences 294* (2015), 456–477.

[63] Mentzer, J. T., DeWitt, W., Keebler, J. S., Min, S., Nix, N. W., Smith, C. D., and Zacharia, Z. G. Defining supply chain management. *Journal of Business logistics 22*, 2 (2001), 1–25.

[64] Mirjalili, S., Mirjalili, S. M., and Yang, X.-S. Binary bat algorithm. *Neural Computing and Applications 25*, 3-4 (2014), 663–681.

[65] Mitchell, M. *An introduction to genetic algorithms*. MIT press, 1998.

[66] Montechiesi, L., Cocconcelli, M., and Rubini, R. Artificial immune system via euclidean distance minimization for anomaly detection in bearings. *Mechanical Systems and Signal Processing* (2015).

[67] Neuweiler, G. Foraging ecology and audition in echolocating bats. *Trends in Ecology & Evolution 4*, 6 (1989), 160–166.

[68] Osman, I. H., and Laporte, G. Metaheuristics: A bibliography. *Annals of Operations research 63*, 5 (1996), 511–623.

[69] Papadimitriou, C. H., and Steiglitz, K. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[70] Parapar, J., Vidal, M. M., and Santos, J. Finding the best parameter setting particle swarm optimisation.

[71] Passino, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems 22*, 3 (2002), 52–67.

[72] Pavani, G. S., de França Queiroz, A., and Pellegrini, J. C. Analysis of ant colony optimization-based routing in optical networks in the presence of byzantine failures. *Information Sciences* (2016).

[73] Picheny, V., Wagner, T., and Ginsbourger, D. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization 48*, 3 (2013), 607–626.

[74] Polakova, R., Tvrdik, J., and Bujok, P. Controlled restart in differential evolution applied to cec2014 benchmark functions. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 2230–2236.

[75] Qin, A. K., Huang, V. L., and Suganthan, P. N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Evolutionary Computation, IEEE Transactions on 13*, 2 (2009), 398–417.

[76] Qin, A. K., Huang, V. L., and Suganthan, P. N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *Trans. Evol. Comp 13*, 2 (Apr. 2009), 398–417.

[77] Ramalhinho Dias Lourenço, H. Supply chain management: An opportunity for metaheuristics. *UPF Economics and Business Working Paper*, 538 (2001).

[78] Ramesh, B., Mohan, V. C. J., and Reddy, V. V. Application of bat algorithm for combined economic load and emission dispatch. *Int. J. of Electricl Engineering and Telecommunications 2*, 1 (2013), 1–9.

[79] Rechenberg, I. *Evolutionsstrategie 94*, vol. 1 of *Werkstatt Bionik und Evolutionstechnik*. Frommann-Holzboog, Stuttgart, 1994.

[80] Rekaby, A. Directed artificial bat algorithm (daba) - a new bio-inspired algorithm. IEEE, pp. 1241–1246.

[81] Schnitzler, H.-U., and O'Dell Jr, W. H. Performance of airborne animal sonar systems: I. microchiroptera. In *Animal sonar systems*. Springer, 1980, pp. 109–181.

[82] Shang, Y.-W., and Qiu, Y.-H. A note on the extended rosenbrock function. *Evolutionary Computation 14*, 1 (2006), 119–126.

[83] Simmons, J. A., Fenton, M. B., and O'Farrell, M. J. Echolocation and pursuit of prey by bats. *Science 203*, 4375 (1979), 16–21.

[84] Stilz, W.-P., and Schnitzler, H.-U. Estimation of the acoustic range of bat echolocation for extended targets. *The Journal of the Acoustical Society of America 132*, 3 (2012), 1765–1775.

[85] Storn, R., and Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization 11*, 4 (1997), 341–359.

[86] Stützle, T. Local search algorithms for combinatorial problems-analysis, algorithms and new applications. *DISKI–Dissertationen zur Künstliken Intelligenz* (1999).

[87] Surlykke, A., Jakobsen, L., Kalko, E. K., and Page, R. A. Echolocation intensity and directionality of perching and flying fringe-lipped bats, trachops cirrhosus (phyllostomidae).

[88] Surlykke, A., and Kalko, E. K. Echolocating bats cry out loud to detect their prey. *PLoS one 3*, 4 (2008), e2036.

[89] Tan, Y., and Zhu, Y. Fireworks algorithm for optimization. In *International Conference in Swarm Intelligence* (2010), Springer, pp. 355–364.

[90] Tanabe, R., and Fukunaga, A. Success-history based parameter adaptation for differential evolution. In *Evolutionary Computation (CEC), 2013 IEEE Congress on* (2013), IEEE, pp. 71–78.

[91] Tanabe, R., and Fukunaga, A. S. Improving the search performance of shade using linear population size reduction. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 1658–1665.

[92] Topal, A. O., and Altun, O. Dynamic virtual bats algorithm (dvba) for global numerical optimization. In *Intelligent Networking and Collaborative Systems (INCoS), 2014 International Conference on* (2014), IEEE, pp. 320–327.

[93] Topal, A. O., and Altun, O. A novel meta-heuristic algorithm: Dynamic virtual bats algorithm. *Information Sciences 354* (2016), 222–235.

[94] Topal, A. O., Altun, O., and Terolli, E. Dynamic virtual bats algorithm (dvba) for minimization of supply chain cost with embedded risk. In *Proceedings of the 2014 European Modelling Symposium* (2014), IEEE Computer Society, pp. 58–64.

[95] Topal, A. O., Altun, O., and Yildiz, Y. E. An effective hybrid of bat algorithm and hill climbing for global optimization of high-dimensional functions. *Journal of Natural and Technical Sciences 20*, 2.

[96] Topal, A. O., Altun, O., and Yildiz, Y. E. Empirical study of dynamic virtual bats algorithm (dvba). In *Proceedings of the 2015 International Scientific Conference-Computer Science'2015* (2015), pp. 236–241.

[97] Topal, A. O., Altun, O., and Yildiz, Y. E. Micro bat algorithm for high dimensional optimization problems. *International Journal of Computer Applications 122*, 12 (2015).

[98] Voss, S., Osman, I. H., and Roucairol, C. Meta-heuristics: Advances and trends in local search paradigms for optimization.

[99] Wang, P., Lin, H.-T., and Wang, T.-S. An improved ant colony system algorithm for solving the ip traceback problem. *Information Sciences 326* (2016), 172–187.

[100] Wang, X., Wang, W., and Wang, Y. An adaptive bat algorithm. In *Intelligent Computing Theories and Technology*. Springer, 2013, pp. 216–223.

[101] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics bulletin* (1945), 80–83.

[102] Xiong, T., Bao, Y., Hu, Z., and Chiong, R. Forecasting interval time series using a fully complex-valued rbf neural network with dpso and pso algorithms. *Information Sciences 305* (2015), 77–92.

[103] Xu, L., Li, Y.-p., Li, Q.-m., Yang, Y.-w., Tang, Z.-m., and Zhang, X.-f. Proportional fair resource allocation based on hybrid ant colony optimization for slow adaptive ofdma system. *Information Sciences 293* (2015), 1–10.

[104] Yang, X.-S. Firefly algorithms for multimodal optimization. In *Stochastic algorithms: foundations and applications*. Springer, 2009, pp. 169–178.

[105] Yang, X.-S. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010.

[106] Yang, X.-S. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*. Springer, 2010, pp. 65–74.

[107] Yang, X.-S. Bat algorithm for multi-objective optimisation. *International Journal of Bio-Inspired Computation 3*, 5 (2011), 267–274.

[108] Yang, X.-S., and Deb, S. Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on* (2009), IEEE, pp. 210–214.

[109] Yang, X.-S., and Hossein Gandomi, A. Bat algorithm: a novel approach for global engineering optimization. *Engineering Computations 29*, 5 (2012), 464–483.

[110] Yashesh, D., Deb, K., and Bandaru, S. Non-uniform mapping in real-coded genetic algorithms. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 2237–2244.

[111] Yilmaz, S., Kucuksille, E. U., and Cengiz, Y. Modified bat algorithm. *Elektronika ir Elektrotechnika 20*, 2 (2014), 71–78.

[112] Yu, C., Kelley, L., Zheng, S., and Tan, Y. Fireworks algorithm with differential mutation for solving the cec 2014 competition problems. In *Evolutionary Computation (CEC), 2014 IEEE Congress on* (2014), IEEE, pp. 3238–3245.

[113] Yuanbin, M., Xinquan, Z., and Shujian, X. Local memory search bat algorithm for grey economic dynamic system. *TELKOMNIKA Indonesian Journal of Electrical Engineering 11*, 9 (2013), 4925–4934.

[114] Zhang, J., and Sanderson, A. C. Jade: Self-adaptive differential evolution with fast and reliable convergence performance. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on* (2007), IEEE, pp. 2251–2258.

CIRRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Topal, Ali Osman

Nationality: Turkish (TC)

Date and Place of Birth: 12.12.1976, Karaman

Marital Status: Married

Phone: 00 355 695374368

Email: aliotopal@gamil.com / aotopal@epoka.edu.al

EDUCATION

Degree	**Institution**		**Year of Graduation**
M.S.	Epoka University	Computer Engineering	2011
B.S.	Gaziantep University	Electrical-Electronic Engineering	2000
High School	Karaman High School		1993

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2000 - 2008	Turgut Ozal Education Company	IT Teacher - Vice Principal
2008 - 2013	Memorial International School of Tirana	IT Teacher - Deputy Director
2013 -	Epoka University	Head of IT Office - Lecturer

PUBLICATIONS

Derived from this thesis

International and National Journals

[1] A. O. Topal and O. Altun, " A novel meta-heuristic algorithm: Dynamic Virtual Bats Algorithm", Information Sciences, 354 (2016): 222-235 - ELSEVIER

[2] A. O. Topal, O. Altun, and Y. E. Yildiz, "An Effective Hybrid of Bat Algorithm and Hill Climbing for Global Optimization of High-dimensional Functions", Journal of Natural and Technical Sciences, vol. 20, no. 2, 2015.

[3] A. O. Topal, O. Altun, and Y. E. Yildiz, "Micro Bat Algorithm for High Dimensional Optimization Problems", International Journal of Computer Applications, vol. 122, no. 12, 2015.

Conference Publications

[4] A. O. Topal, O. Altun, 2014, 'Dynamic Virtual Bats Algorithm (DVBA) for Global Numerical Optimization'. Proceeding of the 6th Intelligent Networking and Collaborative Systems : INCOS 2014. Salerno ITALY. IEEE.

[5] A. O. Topal, O. Altun, E Terolli, 2014, 'Dynamic Virtual Bats Algorithm (DVBA) for Minimization of Supply Chain Cost with Embedded Risk'. Proceeding of the 8th European Modelling Symposium on Mathematical Modelling and Computer Simulation: EMS2014. Pisa ITALY. IEEE.

[6] A. O. Topal, O. Altun, Y. Emre Yildiz, 'Empirical Study of Dynamic Virtual Bats Algorithm (DVBA)'. Proceeding of 7th International Scientific Conference:"Computer Science'2015", Sept.2015, pp. 236-241, DOI: 10.13140/RG.2.1.1364.7844. Durres ALBANIA. (BEST PAPER AWARD)