Integrating Surrogate Modeling to Improve DIRECT, DE and BA Global Optimization Algorithms for Computationally Intensive Problems

by

Abdulbaset Elhadi Saad
G. Diploma. Coventry University 2003
M.Sc. Derby University, 2005

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
in the Department of Mechanical Engineering

Supervisory Committee

Integrating Surrogate Modeling to Improve DIRECT, DE and BA Global Optimization Algorithms for Computationally Intensive Problems

by

Abdulbaset Elhadi Saad
G. Diploma. Coventry University 2003
MSc Derby University, 2005

Supervisory Committee

Dr. Zuomin Dong, (Department of Mechanical Engineering)
Supervisor

Dr. Afzal Suleman, (Department of Mechanical Engineering)
Departmental Member

Dr. Fayez Gebali, (Department Electrical and Computer Engineering)

Outside Member

Abstract

Supervisory Committee

Dr. Zuomin Dong, (Department of Mechanical Engineering)
Supervisor

Dr. Afzal Suleman, (Department of Mechanical Engineering)
Departmental Member

Dr. Fayez Gebali, (Department Electrical and Computer Engineering)
Outside Member

Rapid advances of computer modeling and simulation tools and computing hardware have turned Model Based Design (MBD) a more viable technology. However, using a computationally intensive, "black-box" form MBD software tool to carry out design optimization leads to a number of key challenges. The non-unimodal objective function and/or non-convex feasible search region of the implicit numerical simulations in the optimization problems are beyond the capability of conventional optimization algorithms. In addition, the computationally intensive simulations used to evaluate the objective and/or constraint functions during the MBD process also make conventional stochastic global optimization algorithms unusable due to their requirement of a huge number of objective and constraint function evaluations. Surrogate model, or metamodeling-based global optimization techniques have been introduced to address these issues. Various surrogate models, including kriging, radial basis functions (RBF), multivariate adaptive regression splines (MARS), and polynomial regression (PR), are built using limited samplings on the original objective/constraint functions to reduce needed computation in the search of global optimum.

In many real-world design optimization applications, computationally expensive numerical simulation models are used as objective and/or constraint functions. To solve these problems, enormous fitness function evaluations are required during the evolution based search process when advanced Global Optimization algorithms, such as DIRECT search, Differential Evolution (DE), and Bat Algorithm (BA) are used. In this work, improvements have been made to three widely used global optimization algorithms, Divided Rectangles (DIRECT), Differential Evolution (DE), and Bat Algorithm (BA) by integrating

appropriate surrogate modeling methods to increase the computation efficiency of these algorithms to support MBD. The superior performance of these new algorithms in comparison with their original counterparts are shown using commonly used optimization algorithm testing benchmark problems. Integration of the surrogate modeling methods have considerably improved the search efficiency of the DIRECT, DE, and BA algorithms with significant reduction on the Number of Function Evaluations (NFEs). The newly introduced algorithms are then applied to a complex engineering design optimization problem, the design optimization of floating wind turbine platform, to test its effectiveness in real-world applications. These newly improved algorithms were able to identify better design solutions using considerably lower NFEs on the computationally expensive performance simulation model of the design. The methods of integrating surrogate modeling to improve DIRECT, DE and BA global optimization searches and the resulting algorithms proved to be effective for solving complex and computationally intensive global optimization problems, and formed a foundation for future research in this area.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABC	Artificial Bee Colony
ANN	Artificial Neural Networks
BA	Bat Algorithm
CEBB	Computationally Expensive Black Box
CFD	Computational Fluid Dynamics
DE	Differential Evolution
DIRECT	Dividing rectangles
DOE	Design of experiment
EA	Evolutionary Algorithms
EDO	Engineering design optimization
FEA	Finite Element Analysis
FFA	Firefly Algorithm
FPA	Flower Pollination Algorithm
GA	Genetic Algorithm
GO	Global Optimization
GS	Global Search
GWO	Grey Wolf Optimizer
HEB	High-dimensional expensive black-box
LHD	Latin hypercube design
MARS	Multivariate Adaptive Regression Splines
MBDO	Metamodeling based design optimization
MSE	Mean Square Error
NBGO	Nature -Based Optimization Algorithms
NFE	Number of Function Evaluations
PSO	Particle Swarm Optimization
QRF	Quadratic Response Function
RBF	Radial Basis Function
SQP	Sequential Quadratic Programming
SM	Surrogate Models

Acknowledgements

I would like to express the deepest appreciation to my supervisor Dr. Zuomin Dong for his continuous support, patience, motivation, enthusiasm, and immense knowledge in the course of this work. His kind advice and guidance during my studies kept me constantly engaged with my research. He taught me how to think and how to be independent in research and in everyday life. It has been my greatest honor and pleasure having had the chance to work with Dr. Zuomin Dong.

Supports from the Natural Science and Engineering Research Council of Canada (NSERC), the Clean Transportation Initiative, Transport Canada, and the Libyan-North American Scholarship Program are gratefully acknowledged.

Dedication

A special appreciation is due to my wife and my children, for their understanding, support, and help to foster an inspiring environment of research and learning. Thank you all for your love, support, enthusiasm, and encouragement.

I am greatly indebted to my home country Libya for the scholarship that provided me to finish my PhD. Last but not least, I would like to thank so many of my friends and colleagues at the University of Victoria (UVic) for their assistance and support.

Chapter 1. Introduction

1.1 Background and Motivation

Many optimization problems in engineering, science, and even in medicine can be expressed as global optimization (GO) problems. Thus, over the past three decades, significant progress has been made in introducing and developing efficient and robust GO algorithms. Classical optimization techniques (gradient-based) such as Newton's method, the steepest descent method, and the simple method, are widely used for finding a solution for many optimization problems; however, these classical methods have difficulties with obtaining a global solution particularly when the functions have numerous local minima. Global optimization algorithms can replace the classical optimization methods and, because of their capabilities, have become commonly used when dealing with many challenging and complex engineering optimization problems. Nature-based global optimization (NBGO) algorithms are a branch of the GO methods, and found to be efficient, flexible, and easy to implement. For instance, Genetic Algorithm (GA) [1], Simulated Annealing (SA) [2], and Particle Swarm Optimization (PSO) [3], are advanced GO algorithms that require no gradient information and can provide high accuracy with reasonable time complexity. As well, the NBGO algorithms are capable of solving complex optimization problems and are particularly useful when evaluation of the objective function is inexpensive. Due to the large number of function evaluations typically required by the NBGO methods, it may be inefficient to find a solution when the objective/constrained functions of the problem are computationally expensive. These problems occur in many design areas such as sensitivity analysis, computer simulation, design and control robots, and design of complex mechanical systems. Furthermore, with the real- life engineering design problems becoming more and more complicated, one function evaluation often takes minutes, hours, and even days to be completed by conventional the GO algorithms. Therefore, their applicability to solve computationally expensive real-world engineering applications is limited, since a large number of fitness evaluations are required to reach the region where the global optimum is stabilized. Recently, to address these challenges, surrogate assisted the GO algorithms have been developed and employed to replace the

original computationally expensive black-box (CEBB) functions, thereby promising methodologies for dealing with such computationally expensive optimization problems . The widely-used surrogate models include Quadratic Polynomials, Radial Basis Function (RBF), Kriging, and Neural Networks, etc. [4], which are (statistical) models that are built to approximate the actual function or model. Hence, using such strategies, instead of a high number of function evaluations of the actual system, will reduce the computational cost [5] and will assist the GO methods to converge quickly. Buche *et al.* [6] introduced accelerating evolutionary algorithms with Gaussian process fitness function models to improve their efficiency. Liu *et al.* [7] employed the Gaussian method as a global approximation model to guide the evolutionary algorithm for solving computationally expensive optimization problems by the dimension reduction method. Ratle [8] integrated the kriging-SM as a global search strategy with evolutionary algorithms to reduce the evaluation cost. Ong *et al.* [9] combined an evolutionary algorithm with a SQP solver, in which, during the local search, the RBF surrogate model was employed. Sun *et al.* [10] proposed a two-layer surrogate-assisted particle swarm optimization algorithm where many local surrogate models are employed for fitness approximation. This work introduces a meaningful modification on some of the NBGO to make them more suitable and efficient in handling complex optimization problems. The aim of this work is to use surrogate techniques to accelerate the search efficacy of the selected GO algorithms to reach a global optimum quickly. The combination of the NBGO algorithms with surrogate models can address optimization problems where the objective function evaluation requires computationally expensive simulations. The performance of the proposed algorithms is tested using a number of benchmark functions with different mathematical properties. A real world floating wind turbine platform has been used with seven design variables to show the efficiency and effectiveness of new proposed algorithms. Overall, from the statistical results obtained, it can be seen that the introduced algorithms show a consistent ability to obtain competitive results.

1.2 Research Problem

Optimization problems in engineering design often need computationally expensive computer simulations and analysis to capture the behavior of the expensive black-box system under consideration. Typically, a black-box design problem is a system where no mathematical formula is available, and the system can be represented in terms of its input and output. Computer analysis software, for example, Finite Element Analysis (FEA) and Computational Fluid Dynamics (CFD) are viewed as black-box problems. Continuously increasing computational power enables the development of simulation models that become more and more complex. With increasing model complexity, the computation time of these simulations increases significantly. A problem is considered to be expensive if evaluating a function value is time consuming, which may be related to a complex computer program, the evaluation of the multidisciplinary system, and a huge simulation of FEA /CFD calculation. From an application perspective, there are often restrictions on the variables besides lower and upper bounds, such as linear, nonlinear or even integer constraints. The most general problem formulation is shown in Figure 1.

$$\begin{aligned} &\text{Min} \quad f(x) \\ -\infty \;\le\; x_L \;\le\; x \;\le\; x_U \;\le\; \infty \\ &\textit{Subject to: } b_L \;\le\; Ax \;\le\; b_U \\ &c_L \;\le\; c(x) \;\le\; c_U \\ &x_j \in \mathbb{N} \quad \forall_j \in \mathbb{I} \end{aligned} \tag{1.1}$$

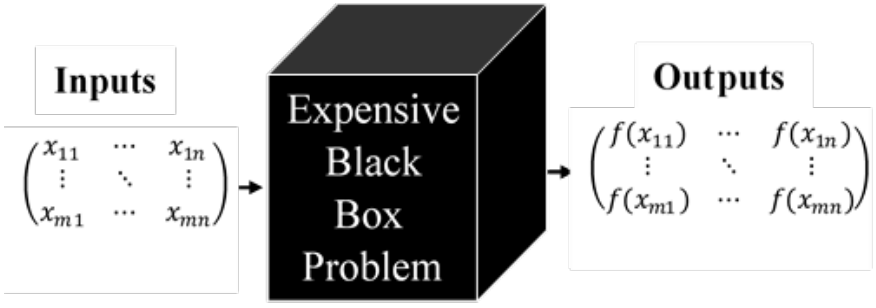


Figure 1: Computationally expensive Black-Box problem

The computationally expensive black-box problem where f (x) ∈ ℝ , and $x_L, x, x_U \in \mathbb{R}^d$. Matrix $\mathbf{A} \in \mathbb{R}^{m_1 \times d}$, $b_L, b_U, \in \mathbb{R}^{m_1}; m_1$ defined as linear constraints and $x_L, x, x_U \in \mathbb{R}^{m_2}$ defines the m_2 as nonlinear constraints. The variables x_i are restricted to be integers, where set $\mathbb{I}$ is an index subset of $\{1, \dots \dots, d\}$. Let $\Omega \in \mathbb{R}^d$ be the feasible set defined only by the simple bounds, the box constraints, and $\Omega_c \in \mathbb{R}^d$ be the feasible set defined by all the constraints as shown in equation (1.1). It is very common to treat all such functions as black-box, when having a set of variables as Input x ∈ $\mathbb{R}^d$ and function value *f(x)* as outputs. This means that no function or derivative information is available, and classic optimization algorithms are not sufficient to solve such a problem. Hence, special the GO solvers are required to solve such optimization problems.

1.3 Research Motivation

Through the last three decades, many global optimization procedures based on nature-inspiration have been effectively established and used to deal practically with different types of global optimization problems. These approaches have shown outstanding search efficiency, ability, and robustness, especially when employed to deal with inexpensive black-box problems. Serious challenges are often faced with the CEBB problems, where the objective function evaluation requires running the computationally expensive simulation model, which may take from several minutes or even hours. Moreover, in many applications, the objective function/constraints are often complex and have numerous local and global minima. Hence, methods that are able to search locally as well as globally need to be improved to find accurate solutions for the problem defined in equation (1.1) within a reasonable number of function evaluations. Consequently, earnest modification of these global optimization methods in order to boost their performance when confronted with computationally expensive simulation engineering design problems is inevitable. As a result, such well-organized search strategies customized for even high-dimensional CEBB optimization problems will explore all the promising regions while keeping the computational resources limited.

FEA and CFD optimization problems are the most challenging engineering tasks in which the complexity of the problem grows with the increasing number of design variables, constraints, and objectives. These issues prevent previously mentioned the GO approaches

from discovering the global optimum solutions quickly. Based on that, they demand increasingly practicable, seamless, and automated integration of adaptive analysis tools and optimization methods. In addition, since the size of the solution space of these problems also increases exponentially with the number of design parameters, any type of dimension reduction can be considered as an exigent need to cope with the difficulty of such missions.

1.4 Objectives of the Research

In this work, the main goal is to provide an efficient optimization algorithm working on CEBB problems. The first step is to investigate the performance and enhance the efficiency of well-known global optimization techniques on the CEBB problems. The key objective of this thesis is not only to focus on the development of global optimization algorithms, but also on the development of combined metamodeling techniques with nature-based algorithms with the intent to improve their overall performance. The goals may be expressed in more detail as follows:

- To study the state of the art in computationally expensive black-box problems (CEBB) design optimization methods.
- To study the behaviours of the existing GO algorithms in handling (CEBB) within a limited time and number of function evaluations.
- To develop a new global optimization strategy that improves the efficiency of different classes of global optimization algorithms.
- To develop a customized surrogate-assisted the deterministic DIRECT search algorithm for solving low dimensional CEBB optimization problems.
- To modify the original Differential Evolution DE optimization method to be well adjusted to solve high-dimensional CEBB optimization problems.
- To improve the capability and efficiency of the Bat algorithm (BA) that can be applied and tested in real-world engineering applications

1.5 Dissertation Outlines

The thesis is structured as follows:

Chapter One gives a general introduction by briefly discussing the motivation, objectives/problem definition, and methodology. The next Chapter presents an overview of the existing GO methods for optimization engineering design problems. The most promising strategy and most well-known surrogate models are identified and their mathematical formulations are described. After a brief review of the background in Chapter Two, Chapter Three describes the optimization algorithm called Kriging-DIREACT search (K-D) advancement for the surrogate assisted sampling-based global search. In Chapter Four, a modification to another stochastic optimization algorithm, named RBF surrogate model guided the DE global optimization algorithm, is discussed. An examination of the proposed algorithm is also conducted in this Chapter, using several representative benchmark functions. Chapter Five presents the most recently developed global optimization algorithms used to solve high-dimension black-box problems; high dimensional benchmark functions are used as well as a real-life case study to examine and investigate their performance. Chapter Six proposes a series of modifications to the Bat Algorithm (BA) used to solve computationally expensive black-box problems. The principal idea is to increase the convergence speed by using Kriging-SM to guide the BA to the most promising region. A real-world engineering case study, Floating Wind Turbine Platform, has been used to examine the robustness of the proposed algorithm. Conclusions are made and possible future directions are suggested in Chapter Six. Finally, a summary of the research and the suggested future work are presented in Chapter Seven.

1.6 Research Contributions

The contributions arising from this work are listed below:

- Carried out an extensive review on GO algorithms, including DIRECT, DE, BA, CS, FFA, FPA, ABC and GWO to identify their pros, cons, and rooms for improvements.
- Intensive study has been achieved on Surrogate Modeling (SM) to identify the appropriate SM to assist advanced GO (Chapter 2).

- Introduced new methodology of integrating Surrogate Modeling with Advanced GO algorithms (Chapter 2).
- Proposed a fast deterministic global optimization algorithm based on the surrogate model (Kriging model) Kriging-DIREACT Search (K-D) algorithm for solving low dimensional complex problems (Chapter 3).
- Developed an efficient GO algorithm for computationally expensive optimization problems by modifying the DE algorithm. Using approximation models with DE (RBF-DE) reduces the burden of these expensive evaluations and directs the DE algorithm to the global solution faster as well as reducing the computation cost. (Chapter 4).
- Tested and compared six mature nature-based GO algorithms using high-dimensional benchmark functions from 30D to 50D with different properties and topology. Compared five optimization algorithms and discussed their results, strengths, and weaknesses in dealing with high-dimensional computationally expensive black-box functions. In addition to closely investigating the effectiveness of chosen optimization techniques, a real-life Floating Wind Turbine Platform in the form of an expensive black-box problem was selected to examine the robustness of the chosen algorithms (Chapter 5).
- Developed and modified the Kriging Bat Algorithm (K-BA) to be used for dealing with computationally expensive black-box problems. This task mainly involved the development of a new surrogate-assisted sampling search technique and the use of appropriate surrogate (s) as a guide to bat optimization method(s). (Chapter 6).
- Applied the developed K-BA method to a real-world engineering problem. The case study was chosen to examine the robustness of the proposed algorithms. The results were then compared with other optimization algorithms. (Chapter 6).
- Summarized the work done in this thesis and made suggestions for future work. (Chapter 7).

Chapter 2. Global Optimization Methods and Surrogate Models

2.1 Introduction

Many challenging optimization problems in engineering, science, and even in economics can be expressed as global optimization (GO) problems. Thus, over the last three decades, a significant amount of time and effort has been spent in developing efficient and robust GO algorithms to deal with the rapidly increasing optimization problems in engineering. Conventional global optimization methods have been developed and shown to be effective and efficient in solving both low and high-dimensional global optimization problems. These methods can be classified into deterministic and stochastic methods. Deterministic algorithms generate a specific sequence of points which converge to a solution, so that different runs result in the same solution. DIRECT search [11] , Branch and Bound [12], and the Clustering method [13] are examples of deterministic algorithms. Deterministic approaches require strong assumptions about the continuity and differentiability of the objective function [14].Therefore, their applicability to real-world applications is limited.

In contrast, stochastic methods use random sampling, so that several runs may result in different solutions for the same problem. The Nature-based optimization algorithms (NBGO), such as GA, SA , Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) are well-known stochastic methods and have shown outstanding performance on many real – world optimization problems, including network design systems [15] , job-shop scheduling [16], travelling salesman [17], power system [18], and training of artificial neural networks (ANNs) [19].

Most of the NBGOs need a large number of function evaluations (NFE) before they reach the global solution or a near-optimal solution, which may make it virtually impossible to apply NBGOs to computationally expensive black-box optimization problems such as fluid dynamic optimization functions. In solving these optimization problems, numerical analysis techniques, such as FEA or CFD simulations, which are frequently involved in evaluating the fitness value of the objective function and solutions, may take minutes, hours, or even days of computation time [20]. Surrogate-assisted,(also known as metamodeling-assisted nature-base methods), such as surrogate-assisted genetic algorithm [21], surrogate-assisted PSO [22], and surrogate-assisted differential evolution [23], have

attracted researchers' attention in recent years. Many problems in the areas of sensitivity analysis, computer simulation, optimal control, and multi-physics, demonstrate the difficulty of reaching a viable solution. As well, design optimization of complex mechanical and control systems requires repeated complex evaluations of system models. Because the computational effort required to construct and use surrogates is usually much lower than that of expensive real-function, to lower the level of complexity and reduce the expensive number of evolutions, surrogate models are employed to replace the original function evaluations for saving computational cost.

The most commonly used surrogate models include polynomial regression (PR), also known as response surface method, Kriging, multivariate adaptive regression splines (MARS), and radial basis functions (RBF). Surrogate models, which aim to model the whole search space, are often utilized in the earlier stages of the research on surrogate-assisted GO algorithms. The use of surrogate models for expensive black-box optimization has become widespread within the last two decades. For example, polynomial and kriging response surface models have been used to solve aerospace design problems [24]. Kriging interpolation was used by Jones *et al.* [25] to develop the EGO method, which is a global optimization method where the next iterate is obtained by maximizing an expected improvement function. Kriging was used in conjunction with pattern search to solve a helicopter rotor blade design problem [26]. Parno *et al.* [27] used DoE with a surrogate model as a stand-in for the expensive objective function within the PSO framework.

2.2 Challenges of real-world engineering design optimization

Computational complexity is a serious issue for the design procedure in the practical engineering design field. Modern advances in computers' simulations, software, and tools are extensively applied in modern engineering design problems. However, due to their computational expense and time consumption the efficiency of the design is reduced. The advancement of both deterministic and stochastic methods that use different constructions has made significant contributions to the development of models and methods utilized to analyze and optimize complex engineering systems for different purposes. In practice, the optimization of any mechanical design problem and improved design efficiency and operability may be considered among the objectives of any optimization method.

Regardless of the optimization objective, any engineering design optimization (EDO) needs knowledge about each stage of the design, design variables and their minimum/maximum limits (called the bounds of variables), constraints, and design performance evaluation models. The first important step in design optimization is to select the accurate optimization method based on the above-mentioned information for a given EDO problem. For example, most real-life or industrial design optimizations are complex in terms of the number of design variables and whether the problems are multidisciplinary, and/or multi-objective. Furthermore, there are typically computationally expensive and/or highly-constrained problems. These issues make the optimization procedure more difficult in both formulating the problem(s) and identifying the solution(s), requiring a highly comprehensive integrated approach. To deal with the computationally expensive problems and the increasing complexity of real-world designs, GO algorithms for optimization have become widely-used. As a result, efficient, robust, and approximate model-assisted GO algorithms can be comprehensively addressed.

From an engineering design perspective, global optimization and efficiency, multi-objectivity/multi-modality, design variable interactions, and costly objective functions are among the major challenges faced by designers today. With an increase in the complexity of a given real-life optimization problem, all these challenges become more serious and can strongly affect the optimization process when attempting to find the best solution. In addition, any design optimization process is limited by computational cost, and the analysis of engineering problems demands many expensive simulation techniques, such as finite element analysis (FEA) and computational fluid dynamics (CFD). As a result, the use of GO becomes an essential process to solve this problem. At this time, performance calculation of a given multi-physical model using computation-intensive analysis like FEA and/or CFD is mostly unavoidable when supporting an EDO process. The evaluation cost associated with these simulations is so extensive that making an assessment of the objective/constraint functions is computationally expensive, demanding minutes, hours, or days of computation time. In engineering, computer simulation and design analysis tools including ANSYS and COMSOL play a significant role in the early stages of the design. As well as being computationally expensive, these models/functions are implicit and unknown to the designer, e.g., black-box functions [28]. Such models provide a set of

output(s) that corresponds to the given input(s), while the designer has no knowledge of their internal structure/expression, making the black box function a significant barrier to design optimization [29]. Based on these facts, this Chapter makes use of effective and efficient optimization strategies to focus on the survey of research work facing such challenges in computationally expensive problems.

2.3 Global optimization approaches and algorithms

Many GO problems involve searching for the global optimal in the design space of the system of interest. The functions to be optimized are often complex, black- box functions with unknown analytical representations, which are hard to evaluate even in the case of non-linear constraints. In order to choose suitable methods for solving a global optimization problem, designers need a thorough comparison of methods, but often the available information is not sufficient. The designer's work is even more difficult because GO methods have two different structures; stochastic or deterministic search mechanisms. Stochastic approaches cannot guarantee that the global optimal will be found in a single run, but the stochastic convergence theory states that the global solution will be identified in a reasonable time. Deterministic optimization methods refer to approaches where a sequence of mathematical calculations is followed and no random search is applied. Deterministic approaches ensure that after a number of iterations, an approximation of the global solution will be reached. A noticeable feature is that the convergence rate to the global solution is much faster compared to the stochastic approaches.

In applied mathematics or numerical analysis, global optimization (GO) algorithms seek the best global variable(s) of a function or model (or a set of functions/models) to be minimized or maximized. These functions are subject to some constraints in the presence of multiple local optima. Such optimization tasks are formulated below:

$$\begin{aligned} &\text{Min } f(x_1, x_2, \dots\dots, x_n) \\ &\text{Subject to:} \\ &\quad g_i(x) \leq 0 \\ &\quad h_j(x) = 0 \\ &x^l \leq x \leq x^u \ (\ or\ x \in S \subseteq R^n\ ) \end{aligned} \tag{2.1}$$

where ***f*** is the function to be optimized, ***f**** is the optimum value, ***x*** is a design variable, ***g*** and ***h*** are constrained functions, ***l*** and ***u*** are respectively lower and upper bounds, and ***S*** is the search space domain. The surface of the objective functions often differ from each other. Functions that need to be optimized but have only one peak or valley are known as unimodal functions, as shown in Figure 2, and those that have many peaks and valleys are known as multimodal functions, as shown in Figure 3. Many multimodal optimization problems that were considered difficult to solve and were intractable even in recent years can now be successfully solved using one of the advanced GO methods.

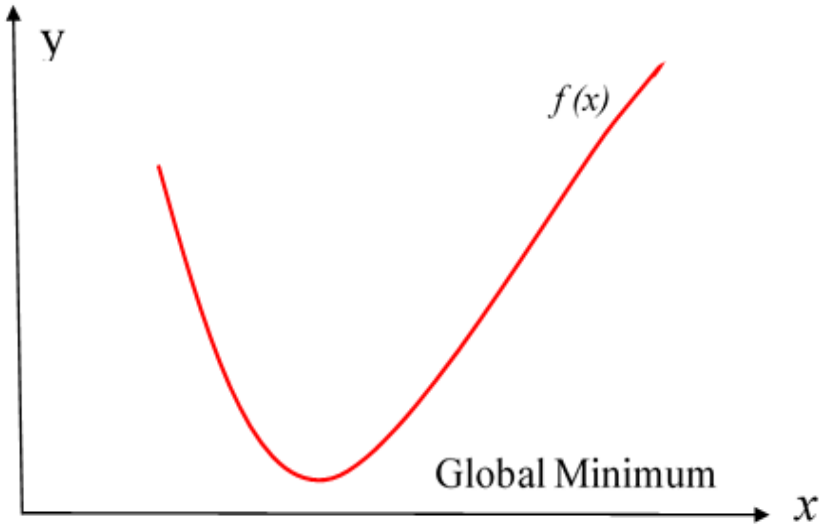


Figure 2.Unimodal Optimization Problem

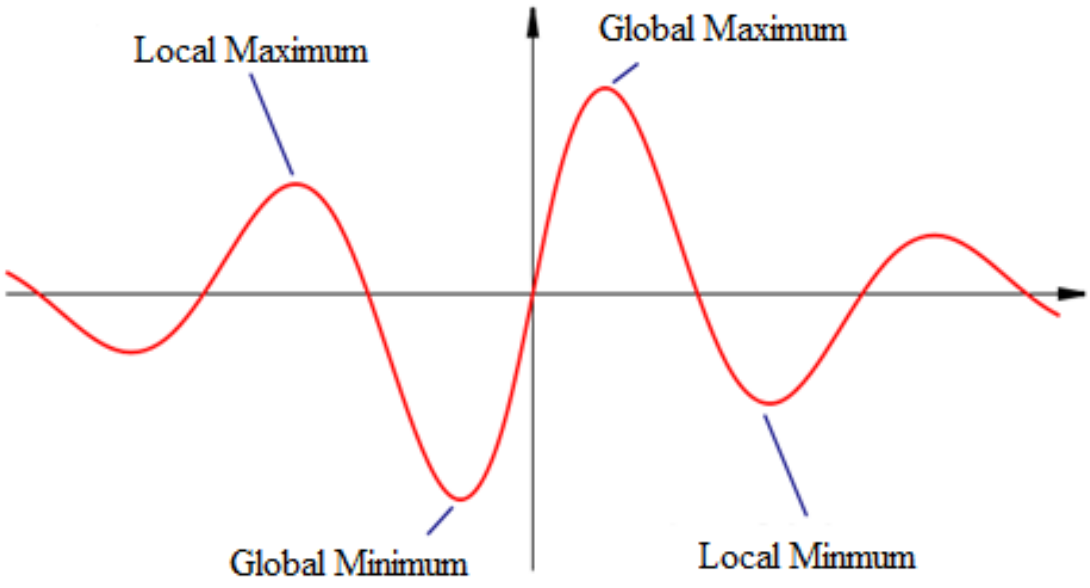


Figure 3.Multimodal Optimization Problem

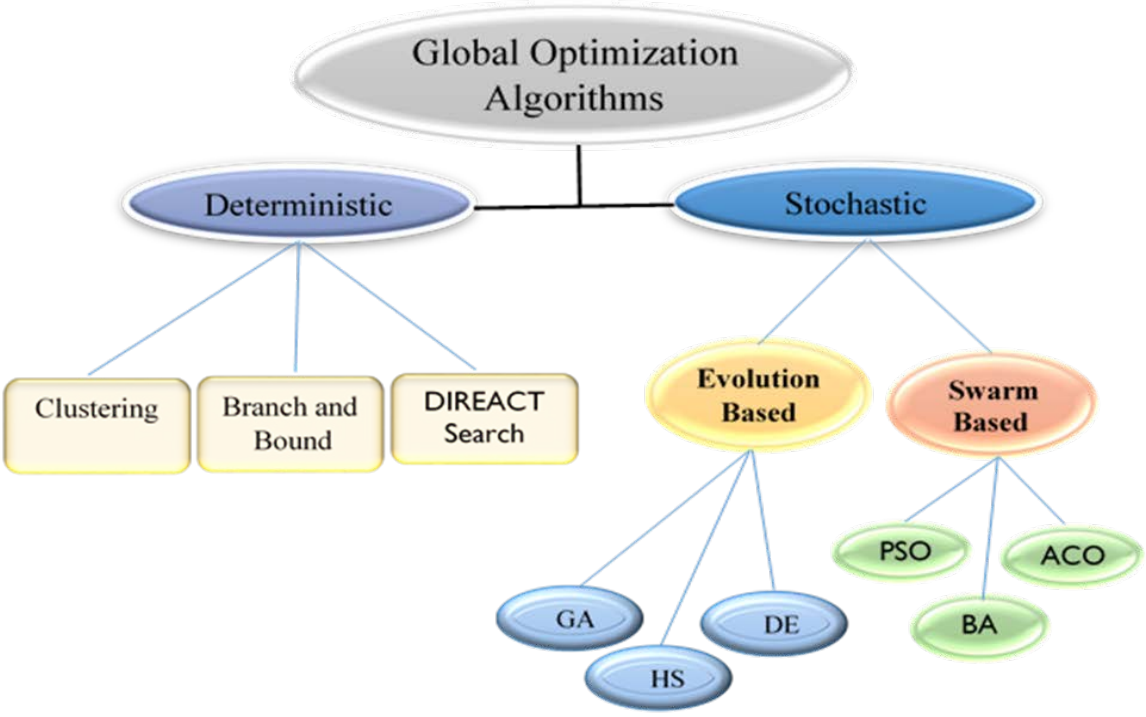


Figure 4. Classification of global optimization approaches

As shown in the Figure 4, GO methods can be categorized into two main classes: stochastic and deterministic approaches, which are briefly reviewed in the following sections. Stochastic methods, for example, use random sampling; hence, different runs might result in different outcomes for an identical problem. On the other hand, deterministic methods work based on a predetermined sequence of point sampling, converging to the global optimum; therefore, different runs result in identical answers for the same optimization problem.

2.3.1 Nature- Based (Stochastic) Global Optimization Algorithms

The nature-based global optimization methods use a random set of sampled points for performing nonlinear search procedures. Due to this randomness, there is no guarantee for optimum attainment within a limited time or computation. The stochastic method is classified into two approaches: evolution based and swarm based algorithms as presented in Figure 4. In the evolution approach, optimization processes start from several different random initial points known as initialized population. The population then updates across numerous generations. In each generation, promising candidates are chosen to become parent candidates. They crossover with each other to generate new candidates, called offspring candidates. Randomly selected offspring candidates are subsequently subject to certain mutations. After that, the approach selects the candidate's solution for the next generation according to the survival selection mechanism of the algorithm. Swarm based algorithms also start with a randomly initialized population size of simple agents. The agents follow very simple rules by communicating locally among themselves and their environment, with no central control to allow globally interesting behaviour to emerge. Among the local optima achieved by this process, the most efficient one is called the global optimum. Swarm based algorithms are a branch of stochastic algorithms and found to be very powerful and mature methods.

Because of the random nature of the algorithm's search and the limitation of the sampling size, stochastic algorithms typically outperform in inexpensive black-box problems while their performance deteriorates considerably, as the problems become computationally expensive. To avoid large computational evaluations in stochastic NBGO methods, the idea of surrogate models should be considered to assist GO methods to converge quickly.

Nonetheless, computational efficiency of these optimization algorithms is always among the main concerns in such procedures. In terms of the computational level, typical stochastic optimization methods include metaheuristic methods, such as nature-inspired and population-based algorithms, involving evolutionary as well as swarm intelligence techniques, as well as simple heuristic methods. Over the years, the GO methods, which are based on nature, are used extensively for different EDO problems. These methods employ different mechanisms and operators to search for individuals that best adapt to the environment. In other words, individuals with a higher level of fitness have a greater chance of surviving and continuing in the optimization process.

Genetic Algorithm

GA is a random search algorithm that is inspired by natural evolution. The algorithm starts with an initial set of points which are collectively known as the population size. The algorithm has a fitness function that is used to calculate a function value of each point (candidate). The fitness value depends on how well the candidate solution solves problems and is the parameter that evaluates a candidate's rank in the movement towards the global optimal solution. One or two candidates are chosen from the population to perform a combination at each stage. The recombination operations are of two types: crossover and mutation. In the first type, two candidates undergo crossover, whereas in mutation, only one candidate takes part. The crossover operation does a randomized exchange between solutions, with the possibility of generating a better solution from a merely adequate one. This operation tends to narrow the search and move towards the global solution. On the other hand, mutation involves flipping possible solutions or an entity in a solution which expands the search exploration of the algorithm. Crossover and mutation rate are the probabilities at which the respective operations are performed [30]. The choice of these probability values reflects the trade-off between exploration and exploitation (or convergence). A higher mutation rate, for example, leads to better exploration but can delay convergence. Moreover, a high crossover rate can lead to faster convergence but may get trapped in a local minimum. Typically, recombination gives an opportunity to reach new and better performing solutions, which are then added to the population. Members in the population that have poor fitness values are thus gradually eliminated. This process is

repeated until either a population member has the desired fitness value, hereby finding a solution, or the algorithm exceeds the time allocated to it, and is terminated.

GA has attracted the interest of many researchers as an effective approach to solve complex structures and achieve better performance. Croce *et al.* [31] presented a GA for solving job shop scheduling problems (JSSPs) with an encoding scheme that was based on preference rules. Sun *et al.* [32] developed a modified GA with a clonal selection and a life span strategy for the JSSPs; the developed algorithm was able to find 21 best known solutions out of 23 benchmarked instances. Lee and Yamak [33] proposed a GA with a new representation scheme that was based on operation completion time and its crossover was able to generate active schedules. Liu *et al.* [34] presented a GA with an operation-based representation and a precedence preserving order-based crossover for the job shop scheduling problems (JSSPs). Zhou *et al.* [35] developed a hybrid algorithm with a new representation scheme called random keys encoding. In this algorithm, a GA was used to obtain an optimal schedule, and then a neighbourhood search was introduced to perform local exploitation and increase the solution quality obtained from the GA. Results showed that the hybrid framework performed better than a GA and heuristic alone. Asadzadeh and Zamanifar [36] proposed a GA that was implemented in parallel, using agents that were also used to create initial populations. Yusof *et al.* [37] developed a hybrid micro-GA that was implemented in parallel for the JSSPs. This algorithm was a combination of an asynchronous colony GA that consisted of colonies with a small number of populations and an autonomous immigration GA with subpopulations. Mahdi *et al.* [38] proposed a hybrid method by integrating three different surrogate models into a GA, where the surrogate models were updated at each iteration of the optimization process. The suitability of each model was then illustrated by comparing the best obtained solution at each iteration.

In particular, GAs perform well for locating global optimization solutions - especially where the optimization problem is inexpensive. Furthermore, GAs can be used in both unconstrained and constrained optimization problems. However, GAs have a slow convergence speed even on the simple optimization problems and require high computation time and a large number of function evaluations.

Simulated Annealing

The SA method is the most popular GO search approach and draws its name from the metallurgical process, “Annealing”. Annealing is a process used to change the properties of a metal wherein the metal is heated to a certain high temperature and then allowed to slowly cool with a specific cooling rate [39]. Similarly, SA explores the design space controlled by two parameters: T temperature and α-cooling rate. The method starts by choosing a random point as its current solution from the given initial set, and the parameter temperature T is given a high value. The method explores the current solution and evaluates it by comparing its value with the current solution. A neighboring solution is improved by using some serious adjustment to the current solution. The value of a current solution determines how effective the current solution is as a potential solution for a given problem. The probability of accepting a current solution depends on its value and the variables T and α, primarily when T is high; the probability of accepting a current solution with an unacceptable value is also high, thus expanding the search space for finding the global solution. As the method proceeds and depending on the cooling rate α, T is gradually decreased; hence, the probability of accepting a weak solution also decreases. If a current solution is accepted, then the approach evolves a new solution from the accepted current solution for the next iteration. This process continues until the global optimal solution is obtained or the stopping criteria are met [39]. In order to improve the SA performance, many researchers have proposed different strategies like faster annealing schedules [40], simulated annealing with an adaptive non-uniform mutation (non-SA) [41], adaptive simulated annealing (ASA) [42], implementation as distributed algorithms [43], hybridization of SA with genetic algorithms [44], integrated SA with support vector machine [45], and finally a combination of SA with artificial neural network [46]. Singh *et al.* [47] proposed hybrid SA with surrogate models to improve the constrained multi-objective SA. The resulting algorithm is referred to as Surrogate Assisted Simulated Annealing (SASA).

In practice, SA has successfully solved the famous traveling salesman problem (TSP). SA has been found to be very powerful for several types of optimization problems; however, the most noticeable drawbacks are that this algorithm requires a long time to find the global

optimum solution, and the trade-off between the global solution and the CPU time needed to obtain the global solution is very high.

Ant Colony Optimization

The ACO algorithm is another random search method that simulates the food searching behavior of ants in real life. It was established in the attempt to realise an optimal solution based on ants' behaviour to find the shortest way between their colony and a source of food as shown in Figure 5. Ants commonly use pheromones as a chemical language to communicate. The ants move based on the amount of pheromones - the richer the pheromone trail on a path, the more likely it would have been previously tracked by other ants. So, in all probability, a shorter trail has a higher amount of pheromone, and ants will tend to choose that shorter route between the food location and their colony. The process that ants use is described and translated as a global optimization algorithm tool for solving optimization problems. The algorithm was designed primarily for discrete variable optimization problems although it has been used for solving continuous optimization problems and other problems as well [48]. Since its presentation in 1992, many different ACO methods have been introduced, including ant colony system (ACS) [49] and MAX-MIN ant system (MMAS) [50]. Meanwhile, ACO approaches have been widely studied and effectively applied to solve multi-objective problems such as travelling salesmen problem (TSP) [51], portfolio selection problem [52], vehicle routing problem [53], scheduling problem [54], and network optimization problem [55].

ACO is one of the most successful methods among swarm intelligence algorithms and has been effectively used in many real-life application problems. Although ACO has a powerful ability to converge to the optimal solutions of many optimization problems, there is a high possibility of getting trapped in local optima; as well, the convergence speed of ACO is very slow [55].

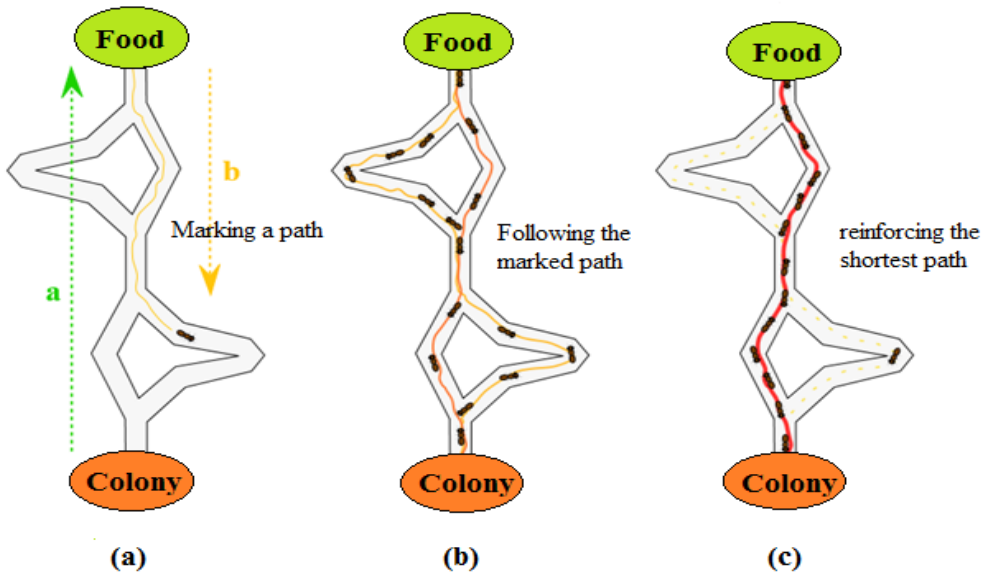


Figure 5. Ant Colony Optimization Algorithm Process [17]

where *N* and *S* denote Nest and Source, *a* is ongoing direction and *b* is returning direction. Sub Figure 5 (a) shows the early process where ants start finding a path between nest and source and lay pheromone. Figure 5 (b) shows an intermediate process where ants go through all possible paths. Figure 5 (c) shows that most ants choose the path with highest pheromone.

Particle Swarm Optimization

Among random search approaches is the Particle Swarm Optimization (PSO). This method was first introduced in 1995 [56]. PSO is a robust and mature global optimization technique based on the movement and intelligence of swarms to guide the particles to search for globally optimal solutions. PSO is inspired by the ability of flocks of birds, schools of fish, or packs of animals to adapt to their environment, find sources of food, and avoid predators by implementing the "sharing information" method; hence, developing an evolutionary advantage as it can be seen in Figure 6 . In the PSO optimization process, a set of randomly generated solutions spreads in the search space towards the optimal solution over a number of repetitions based on a large amount of information about the search space. This set of particles is assimilated and shared by all members of the group. In PSO, each agent or particle is a solution and the best value among those solutions is considered to be the global

solution after the stopping is satisfied. PSO has attracted the attention of many scientists' due to its ability to search very large design spaces and make few assumptions about the optimization problem being optimized. Valdez *et al.* [56] presented an improved version of the PSO method by combining the advantages of PSOs and GAs. Eberhart and Shi [57] proposed a modified PSO, which can find the optimal solution in a dynamic environment. In [58], a hybrid PSO with a wavelet mutation (HWPSO) was given, in which the mutation incorporates with a wavelet function. The success of this PSO was utilized to increase its efficiency by adapting an inertia weight strategy [59]. Based on the success of mutation mechanisms and different local search techniques, a superior solution guided PSO (SSG-PSO) was introduced [60]. Through using Cauchy mutation, a hybrid PSO (HPSO) was presented [61]. Two modified PSOs were introduced based on the second personal best and the second global best particle [62]. A modified PSO was presented to avoid premature convergence using parameter automation strategy [63]. In order to avoid being trapped in local optima in the convergence process, other improved PSOs have been proposed, such as orthogonal learning strategy [64] and elitist learning strategy [65]. Regis [22] developed an RBF surrogate model assisted PSO to solve expensive black-box problems by generating multiple trial velocities and positions for each particle in each iteration and then the RBF surrogate model was used to select the most promising trial position for each particle.

PSOs have been successfully applied to optimize various continuous nonlinear functions. Although the applications of PSOs on computationally expensive optimization problems are still limited, PSOs have certain advantages such as easy implementation and high efficiency, and they do not need the gradient information for optimization of the problem. However, as PSOs can become trapped in local optima when handling complex multimodal functions, the convergence speed and being trapped in the local optima are considered to be major drawbacks. Continuous research is being carried out to address such disadvantages in this well- known and widely used GO optimization algorithm.

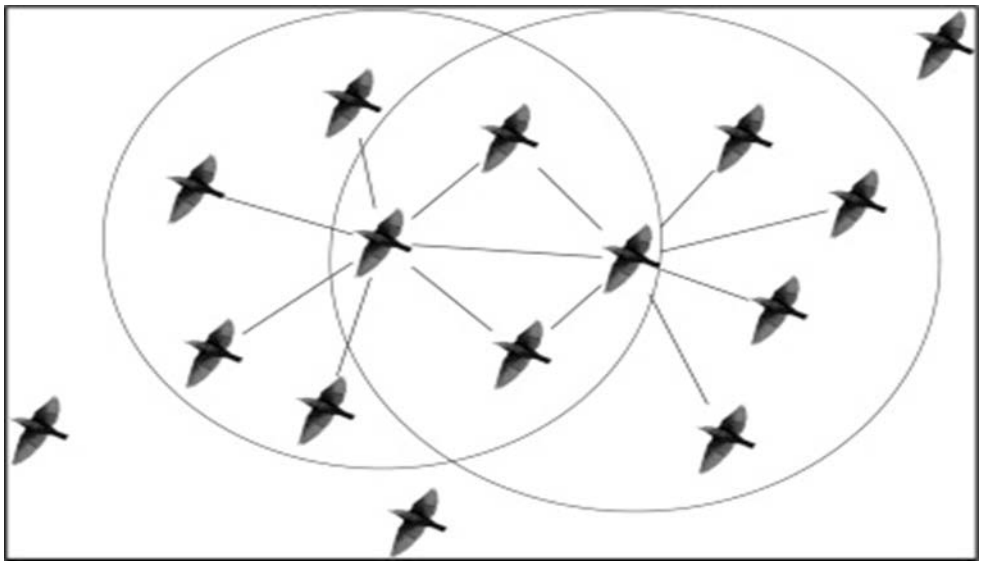


Figure 6. Inspiration of Particle Swarm Optimization [18]

Differential Evolution

Differential evolution (DE) is one of the most powerful evolutionary GO algorithms used at present. DE functions use the same computational steps as the evolutionary algorithms (EA). The DE is considered to be similar to GA since it uses comparable operators; crossover, mutation, and selection. The main difference between DE and GA is that DE is based on mutation operation while GA is based on crossover operations to select a better solution. This method was presented by Storn and Price in 1997 [66]. Since this approach is based on mutation, it employs the mutation as a search tool and takes advantage of the selection process in order to direct the search towards the promising regions in the design space. Target Vector, Mutant Vector, and Trail Vector are the three main steps that DE applies for creating a new population in each iteration. The target vector is the vector that holds the candidate solution for the design space; the mutant vector is the mutation of the target vector; and the trail vector is the outcome vector after applying the crossover process between target vector and mutant vector. DE begins with population initialization followed by evaluation to define the fittest candidates of the population as shown in Figure 7. Then, the mutation operator is applied and new parameter vectors are created by adding the weighted difference of the two population vectors with the third vector. Later, the crossover operator is used, and the DE reaches a final stage of selection.

Due to its flexibility and simplicity, the DE has been commonly used in many engineering applications. The characteristics that make DE a powerful algorithm is its capability of handling non-differentiable, nonlinear, and multimodal objective functions. It has been used to train neural networks having real and constrained integer weights [66]. There are a handful of results showing that the DE is often more effective and more efficient than genetic algorithms while it suffers from slow convergence or being trapped in local optimum.

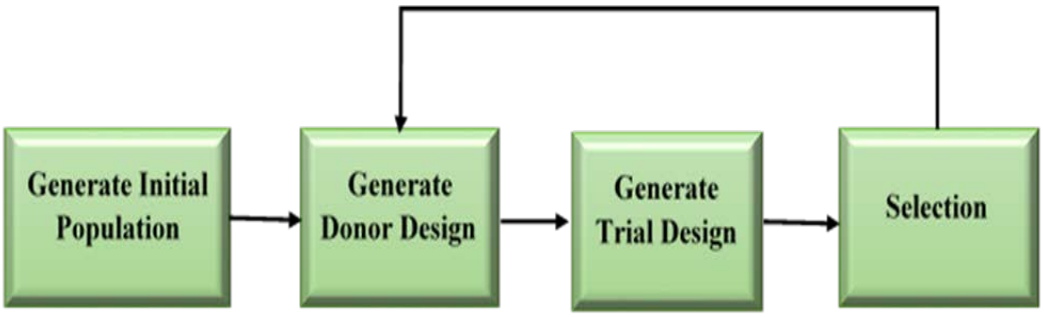


Figure 7. Main Steps of Differential Evolution Algorithm

2.3.2 Conventional Deterministic Global Optimization Methods

Deterministic or mathematical optimization algorithms such as Branch and bound, DIRECT Search, Clustering, and Tunnelling are the classic branches of optimization algorithms in mathematics. They solve an optimization problem by generating a deterministic sequence of points converging to an optimal solution. These methods will reach the same solution in different runs as compared to other stochastic search GO algorithms and converge quickly to the global optimum. Furthermore, where the starting point, bounds of design variables, and termination criteria are fixed, the number of fitness function evaluations will be the same for different trials for a given problem. However, many of the deterministic methods need mathematical equations of the optimization problem to have the function values - rather than their derivatives. The need for these equations may be considered the major weakness of such methods. Deterministic optimization algorithms show outstanding performance for expensive black-box problems but have difficulty with high dimensionality problems.

DIRECT Search Method

Dividing rectangles (DIRECT) search, as an efficient and robust deterministic global optimization algorithm based on objective-oriented sequential sampling. DIRECT algorithm is one of the most recognized GO methods that can find the global optimum for specific optimization problems without requiring any gradient information of the objective function. The algorithm operates through a process of dividing the search space into to a number of rectangles - one of its unique features. As a modification of the original Lipschitzian method, the algorithm eliminates the need to specify a Lipschitz constant by carrying out intensive searches using all possible constants from zero to infinity. The first step in the DIRECT algorithm is to divide the search space into the unit hypercubes, as shown Figure 8. The algorithm then samples the value of the function at the center of each hypercube, instead of computing it at the summits, starting at the center of the focused space. Subsequently, at each iteration, DIRECT search selects and subdivides the set of hypercubes that are most likely to contain the lowest value of the objective function. The center point of each cube will be considered as the starting point by DIRECT while it searches for the best value of the objective function within. DIRECT search algorithm is very common because it is easy to implement and can be applied to many nonlinear optimization problems, particularly when the derivatives are expensive or unavailable.

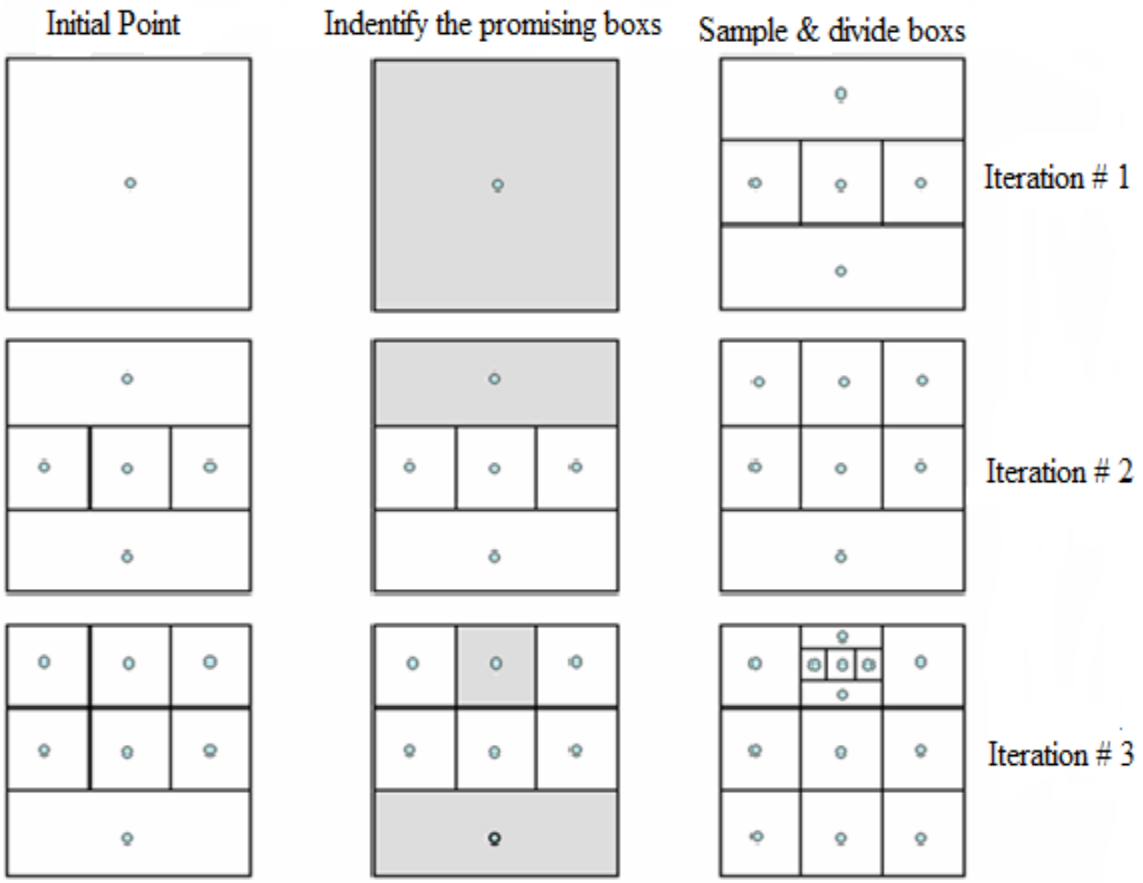


Figure 8. Optimization process for three iterations in DIRECT algorithm

Branch and bound algorithm

The Branch and Bound (B&B) algorithm was introduced by Land and Doig as a generic algorithm [67]. The Branch and Bound has been presented for finding the optimal solution of real-world discrete programming problems and for large-scale optimization problems in particular. This approach has three main steps: selection of the node to process, bound calculation, and branching [67]. By estimating the upper and lower bounds of the design variables, a B&B paradigm involves a systematic evaluation of the entire search space of possible candidates. As the procedure is repeated, many large subsets of fruitless candidates are discarded. B&B terminology, a general description, examples, and other details, are found in [12, 68].

2.4 Computationally Expensive Black Box Problems and SM

Black-box problems, computational cost, and not having mathematical expressions are the three main issues in advanced optimization problems, so their combination makes the problem very challenging [28]. Surrogate models or approximation techniques are most commonly utilized for replacing expensive black-box models such as FEA and/or CFD models to reduce the computational cost in the function evaluation of the optimization process. Moreover, as well as being computationally expensive, most engineering design analysis processes do not render explicit mathematical functions; they are often referred to as black-box functions. In other words, key information such as functional form, non-linearity of the function, and variable correlations are unknown to the designer.

Surrogate models are constructed to replace computational intensity by approximating the original computational expensive black-box function as a cheap model white-box function as shown in Figure 9. The main concept of surrogate models is to apply the DOE's results to build an approximation model over the search space. Least square regression, Kriging, multivariate adaptive regression splines (MARS), polynomial regression (PR), and radial basis functions (RBF) are surrogate techniques which can be used to replace the real system based on a set of expensive sampling points. Surrogate models can be generated by a mathematical formula based on a set of points from a computer simulation and an analysis of the actual system. The first step of building surrogate model starts by generate a population of solutions using the Latin hypercube design (LHM), and then calculates the fitness values of all points, based on the true fitness function and hence update the fitness function. The surrogate model is then built to approximate the actual model as shown in Figure 10. To lower the above-mentioned challenges in modeling and design optimization of computationally expensive black-box (CEBB) problems, surrogate models have been used to assist GO algorithms to converge quickly and reach the global optimum with a low number of function evaluations. The approximate function built by a surrogate model is simpler than the original function known as "cheap function", so the computational complexity is reduced.

A surrogate model is an approximation of a simulation used to construct simpler and lower computational cost models; if the original simulation is represented as $f(x)$ and the metamodeling is represented as $\hat{f}(x)$, then, $\hat{f}(x) = f(x) + \epsilon(x)$, where $\epsilon(x)$ is the

approximated error. The internal behavior of $f(x)$ does not need to be known (or understood); only the input/output behaviour is important. A model is constructed based on the response of the simulator to a limited number of intelligently chosen data points. Surrogate model generate simpler representations that capture relations between the relevant information of the input and output variables and not in the underlying process.

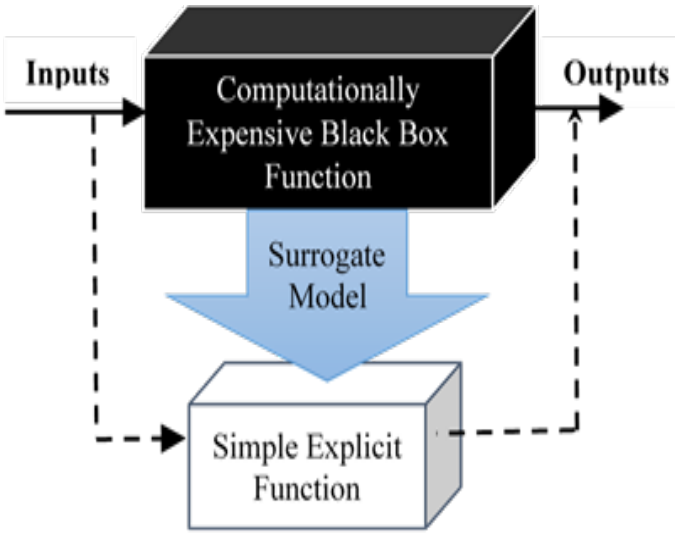
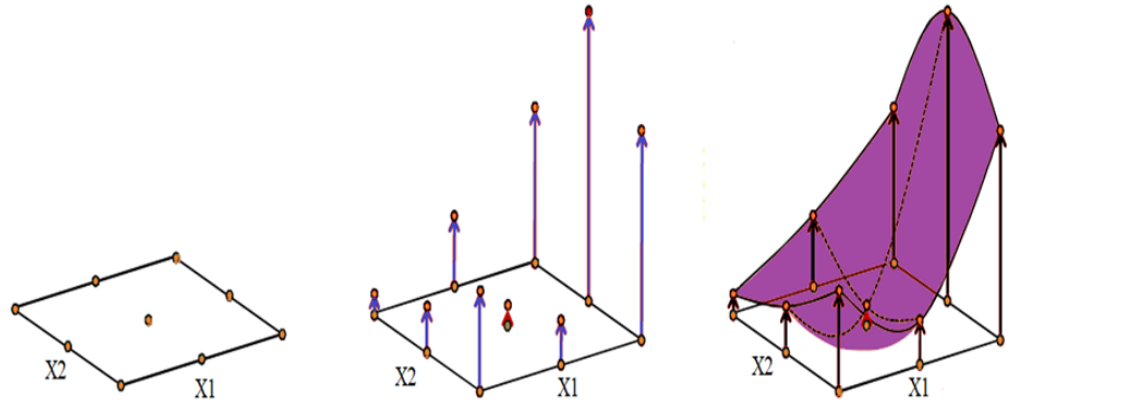


Figure 9. Surrogate Turns Back Box Function to Simple Explicit Function

In order to reach the highest accuracy, it is important to select the appropriate surrogate model to present the actual model. For any optimization problem in which the calculations of the objective function of the problem involve extensive simulation analysis, the metamodeling can reduce computation time, thereby allowing an achievable global optimization solution.

In this research, a surrogate-assisted GO algorithm is proposed that combines a global surrogate model to the objective function with an advanced GO algorithm to speed up the search of global optimum for a computationally expensive problem. In this work, an initial sample is generated utilizing LHM as sampling method. A Kriging model is then used to build a model based on the initial sample points. Once the model is built, GO algorithm is used to maximize the improvement of the search. The best solution obtained from GO algorithm is evaluated on the exact fitness function and then added to the initial samples and a new model is reconstructed.



(a) DOE in the Design Space (b) Function Evaluation (c) Function Prediction by Surrogate Model

Figure 10. Steps of Function Prediction by Surrogate Model

This section also provides a background of the most well-known surrogate methods including: Radial Basis Functions (RBF), Kriging (KRG), Polynomial Regression (PR), and Support Vector Regression (SVR). Surrogate models have been proven to be flexible in implementation and are cheaper to evaluate when compared to the actual system.

2.4.1 Kriging Surrogate Model

Kriging is an interpolation method which has been used in many applications for the estimation of the real system based on a set of designs of experiment (DOE). The Kriging predictor can estimate the function by defining the mean square error (MSE) of the function. Kriging has become popular due to its ability to mimic the behaviour of computationally costly simulation systems. The Kriging method depends on mathematical and statistical models. The addition of a statistical model that includes probability separates Kriging methods from deterministic methods. Kriging defines the correlation model between two points x^1 and x^2 as follows:

$$R\left(\Theta, \boldsymbol{x}^1, \boldsymbol{x}^2\right) = \prod_{j=1}^{n} R_j\left(\theta_j, x_j^1 - x_j^2\right) \tag{2.1}$$

Here, n is the dimension of sample points. If the Gaussian correlation function is employed, it is formulated as:

$$R_j\left(\theta_j, x_j^1 - x_j^2\right) = \exp\left(-\theta_j \left|x_j^1 - x_j^2\right|^2\right) \tag{2.2}$$

Generally, the predictor and the estimated mean squared error (MSE) of the Kriging method can be used to explore the design space. Based on the correlation model introduced above, the predictor and MSE are expressed as follows:

$$\widehat{f}\,(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{r}^{T}(\boldsymbol{x})\boldsymbol{R}^{-1}\,(\boldsymbol{y} - \boldsymbol{1}\hat{\boldsymbol{\mu}}) \tag{2.3}$$

$$\hat{s}^{2}(\boldsymbol{x}) = \hat{\sigma}^{2}\left[1 - r^{T}(\boldsymbol{x})\,\boldsymbol{R}^{-1}\,\boldsymbol{r}(\boldsymbol{x}) + \frac{(1-\boldsymbol{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}))^{2}}{\boldsymbol{1}^{\mathrm{T}}\,\boldsymbol{R}^{-1}\boldsymbol{1}}\right] \tag{2.4}$$

Assume that there are M sample points $s^{(1)}$, $s^{(2)}$, ….., $s^{(M)}$and their true values are evaluated $\boldsymbol{y} = \left[y(\boldsymbol{s}^{(\boldsymbol{1})}), y\left(\boldsymbol{s}^{(\boldsymbol{2})}\right), \ldots, y(\boldsymbol{s}^{(\boldsymbol{M})})\right]^{\boldsymbol{T}}$**.** In equation (3) and (4), $\boldsymbol{R}$ is an $M \times M$ correlations matrix whose (i, j) entry is defined as $R\left(\Theta, \boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)}\right)$ and $\boldsymbol{r}(\boldsymbol{x}) = \left[R\left(\Theta, \boldsymbol{x}, \boldsymbol{s}^{(1)}\right), R(\Theta, \boldsymbol{x}, \boldsymbol{s}^{(2)}, \ldots, R(\Theta, \boldsymbol{x}, \boldsymbol{s}^{(\boldsymbol{M})}\right]^{T}$. Where, Parameters $\hat{\mu}$, $\hat{\sigma}^{2}$, Θ can be obtained by maximum likelihood estimation. The MSE function reflects the uncertainty of the Kriging predictor. Approximation accuracy depends on the distance between untested locations and the given sample points. Typically, the Kriging predictor can perform better if the untested location is closer to the sample points.

2.4.2 Radial Basis Functions (RBF) Surrogate Model

Radial basis functions (RBF), introduced by Hardy [69], are useful for representing irregular surfaces using a combination of basic functions. Radial basis functions depend on the radial distance from specific sample data points to build a surrogate model. RBFs have been successfully applied in many different fields such as function approximation, time series prediction, and control [70].The most common advantage of RBFs is their capability to model curvature well. The RBF model can be expressed as:

$$\widehat{\mathcal{Y}}\,(\boldsymbol{x}) = \sum_{i=1}^{n} \omega_i\,\varphi(\boldsymbol{x}) + P(x), \tag{2.5}$$

where, $\omega_i$are the i unknown weight coefficient, $\varphi(\boldsymbol{x}) = \varphi(\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}\right\|)$are the basis functions that depend on the Euclidean distance between the observed point $x^{(i)}$and the untried point $\boldsymbol{x}$ (similar to the correlation function of kriging model); $P(\boldsymbol{x})$ is the global trend function which is taken as a constant $\beta_0$here. To ensure the function values at

observed points are reproduced by the RBFs predictor, the following constraints should be satisfied:

$$\hat{y}(\boldsymbol{x}^{(i)}) = y^{(i)}, i = 1, \dots, n \tag{2.6}$$

Then the additional constraints for $P(\boldsymbol{x})$ should be imposed as:

$$\sum_{i=0}^{n} \omega_i = 0 \tag{2.7}$$

Solving the liner equations formed by Eq. (2.6) and Eq. (2.7) for ω_i and β_0, and substituting into Eq. (2.5) yields the RBFs Predictor as:

$$\hat{y}(\boldsymbol{x}) = \beta_0 + \varphi^{\mathrm{T}}(\boldsymbol{x})\Psi^{-1}(\boldsymbol{y}_s - \beta_0 \mathbf{1}) \tag{2.8}$$

where $\beta_0(\theta) = (\mathbf{1}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\mathbf{1})^{-1}\mathbf{1}^{\mathrm{T}}\boldsymbol{\Psi}^{-1}\boldsymbol{y}_s$ and $\boldsymbol{\varphi}(\boldsymbol{x})$ are defined as

$$\boldsymbol{\Psi} = \left[\varphi(\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\right\|\right]_{ij} \in \mathbb{R}^{n\times n}, \varphi(\boldsymbol{x}) := \left[\varphi(\left\|x^{(i)} - \boldsymbol{x}\right\|\right]_i \in \mathbb{R} \tag{2.9}$$

where $\boldsymbol{\Psi}$ is the basis-function and $\boldsymbol{\varphi}(\boldsymbol{x})$ is the basis function vector and they are different from the correlation matrix **R** and the correlation vector $\boldsymbol{r}(\boldsymbol{x})$ of the Kriging predictor. To build RBF models, one needs to prescribe the type of basis functions that only depends on the Euclidean distance $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x'}$ between any two sites $\boldsymbol{x}$ and $\boldsymbol{x'}$. Compared to the correlation function used for a Kriging model, more choices are available for a RBFs model, which are partially listed in Table 1.

Table 1: Basis functions for RBFs surrogate model

Basis functions	**Formulations**
Gaussian (GAUSS)	$\phi(r) = e^{-r^2/2\sigma^2} (e.g.\ \sigma^2 = 1)$
Power function (POW)	$\phi(r) = r^{\beta}, 1 \leq \beta \leq 3\ (e.g. \beta = 1.8)$
Thin Plate Spline (TPS)	$\phi(r) = r^2\, ln(r)$
Hardy's Multiquadric (HMQ)	$\phi(r) = \sqrt{1 + r^2}$
Hardy's Inverse Multiquadric (HIMQ)	$\phi(r) = 1/\sqrt{1 + r^2}$

All the basis functions listed in Table 1 can be classified into two categories: decaying functions (such as GAUSS and HIMQ) and growing functions (POW, TPS and HMQ). The decaying functions can yield positive definite matrix $\boldsymbol{\Psi}$, which allows for the use of Cholesky decomposition for its inversion; the growing functions generally result in a non-positive definite matrix $\boldsymbol{\Psi}$ and thus LU decomposition is usually used alternatively.

2.4.3 Quadratic Response Surface Surrogate Model

The Quadratic Response Function (QRF) has been used successfully for decades as an approximation or metamodeling predictor for unknown systems. Initially, it was introduced for the analysis of physical systems based on limited information about the actual system. QRF has been widely used for constructing approximations in a variety of engineering application problems. QRF estimates a problem model by employing the least squares techniques using a set of expensive points in the search space. First and second order polynomials are extensively utilized for approximating functions.

The true quadratic RSM can be written in the following form:

$$\mathcal{Y}(\boldsymbol{x}) = \hat{\mathcal{Y}}(\boldsymbol{x}) + \varepsilon\,, \boldsymbol{x} \in \mathbb{R}^m\,, \tag{2.10}$$

where, $\hat{\mathcal{Y}}(\boldsymbol{x})$ is the quadratic polynomial approximation and ε is the random error which is assumed to be normally distributed with mean zero and variance of σ^2. The error ε_i at each observation is supposed to be independent and identically distributed. The quadratic RSM predictor $\hat{\mathcal{Y}}(\boldsymbol{x})$ can be defined as:

$$\hat{\mathcal{Y}}\,(\boldsymbol{x}) = \beta_0 + \sum_{i=1}^{m} \beta_i\, x_i + \sum_{i=1}^{m} \beta_{ii}\, x_i^2 + \sum_{i=1}^{m} \sum_{j \geq i}^{m} \beta_{ij}\, x_i x_j\,, \tag{2.11}$$

where, β_0, β_i, $\beta_{ii}$and β_{ij} are unknown coefficients to be determined. Since there are totally p = (m + **1**) (m+ **2**) /2 unknown coefficients in Eq. (2.11), building a quadratic RSM with m variables requires at least p sample points. Let $\boldsymbol{\beta} \in \mathbb{R}^p$ be the column vector which contains these p unknown coefficients. The least square estimator of $\boldsymbol{\beta}$ is:

$$\boldsymbol{\beta} = (\mathrm{U}^{\mathrm{T}}\mathrm{U})^{-1}\mathrm{U}^{\mathrm{T}}\, \boldsymbol{y}_s \tag{2.12}$$

Where:

$$\mathrm{U}=\begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_m^{(1)} & x_1^{(1)}x_2^{(1)} & \cdots & x_{m-1}^{(1)}x_m^{(1)} & (x_1^{(1)})^2 & \cdots & (x_m^{(1)})^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_m^{(n)} & x_1^{(n)}x_2^{(n)} & \cdots & x_{m-1}^{n}x_m^{(n)} & (x_2^{(n)})^2 & \cdots & (x_m^{(n)})^2 \end{bmatrix} \in \mathbb{R}^{n\times p}$$

After the unknown coefficients in $\boldsymbol{\beta}$ are determined, the approximated response $\hat{y}$ at any untried $\boldsymbol{x}$ can be efficiently predicted by Eq. (2.11).

2.5 Surrogate Models Assist GO Algorithms

Surrogate-assist GO methods represent a class of optimization methodologies that make use of surrogate modeling techniques to help GO algorithms quickly find the global optima. They provide a novel optimization framework in which the conventional optimization algorithms are used for sub-optimization. Surrogate modeling techniques are of particular interest for engineering design when expensive analysis codes (e.g. Computation Fluid Dynamics (CFD) or Finite Element Analysis (FEA)), are used. They can be employed to greatly improve the search efficiency and are very helpful in finding global optima and integrating simulation codes of different disciplines in the optimization process Figure 11. For optimization problems, surrogate models can be regarded as approximation models for the cost function (s) and constrain function (s), which are built from sampled data obtained by randomly probing the design space (called DOE. Once the surrogate models are built, an optimization algorithm such as an evolutionary algorithm (EA) can be used to search the new design (based on the surrogate models) that is most likely to be the optimum. Since the prediction with a surrogate model is generally much more efficient than with a numerical analysis, the computational cost associated with the search based on the surrogate models is generally negligible. Surrogate modeling is referred to as a technique that makes use of the DOE set of expensive points to construct surrogate models. These models are sufficient to predict the output of an expensive computer simulation at untried points in the design space. Thus, how to choose sample points, how to build surrogate models, and how to evaluate the accuracy of surrogate models are key issues for surrogate models to find the optimal values of the design variables for such CEBB problems. Indeed, a CEBB of design variables presents an exponential difficulty for both problem modeling and optimization [71].

As Wang and Shan stated in [72], there are three different types of metamodeling-based design optimization MBDO including sequential, adaptive, and direct sampling approaches. The first is the most popular and traditional MBDO, using the surrogate as a surrogate of the expensive function where new sample points are not available because of

limitations on budget and/or time. In the second approach, to improve accuracy, the designer may employ additional sample points to successively update the surrogate model. In the third and more recent approach, the surrogate models are only used as guidelines to assist the global optimization method in generating new sample points toward reaching a highly efficient global optimum solution [73]. With respect to optimization processes, computational efficiency is affected by the cost of both function evaluation and the optimization process, depending on the nature of the CEBB problem at hand. The basis of surrogate-assisted optimization is to employ one/several surrogate model(s) of the objective function f to enhance the efficiency of the search in terms of the number of function evaluations required according to the stopping criterion. The basic assumption is that the evaluation of fitness function is relatively expensive in terms of time and/or money –and the learning/building of surrogate model(s) is inexpensive. Generally speaking, the framework of surrogate -assist GO algorithms consists of four common steps:

- Generating a set of sample points in the design space using the LHM technique
- Building surrogate model(s) from a training set of previously evaluated expensive points
- Using the best points of the model(s) as the initial population in an optimization process
- Computing additional points; and
- Updating the set sampling points.

The main steps of surrogate models or metamodeling which assist global optimization algorithms are presented in Figure 11.

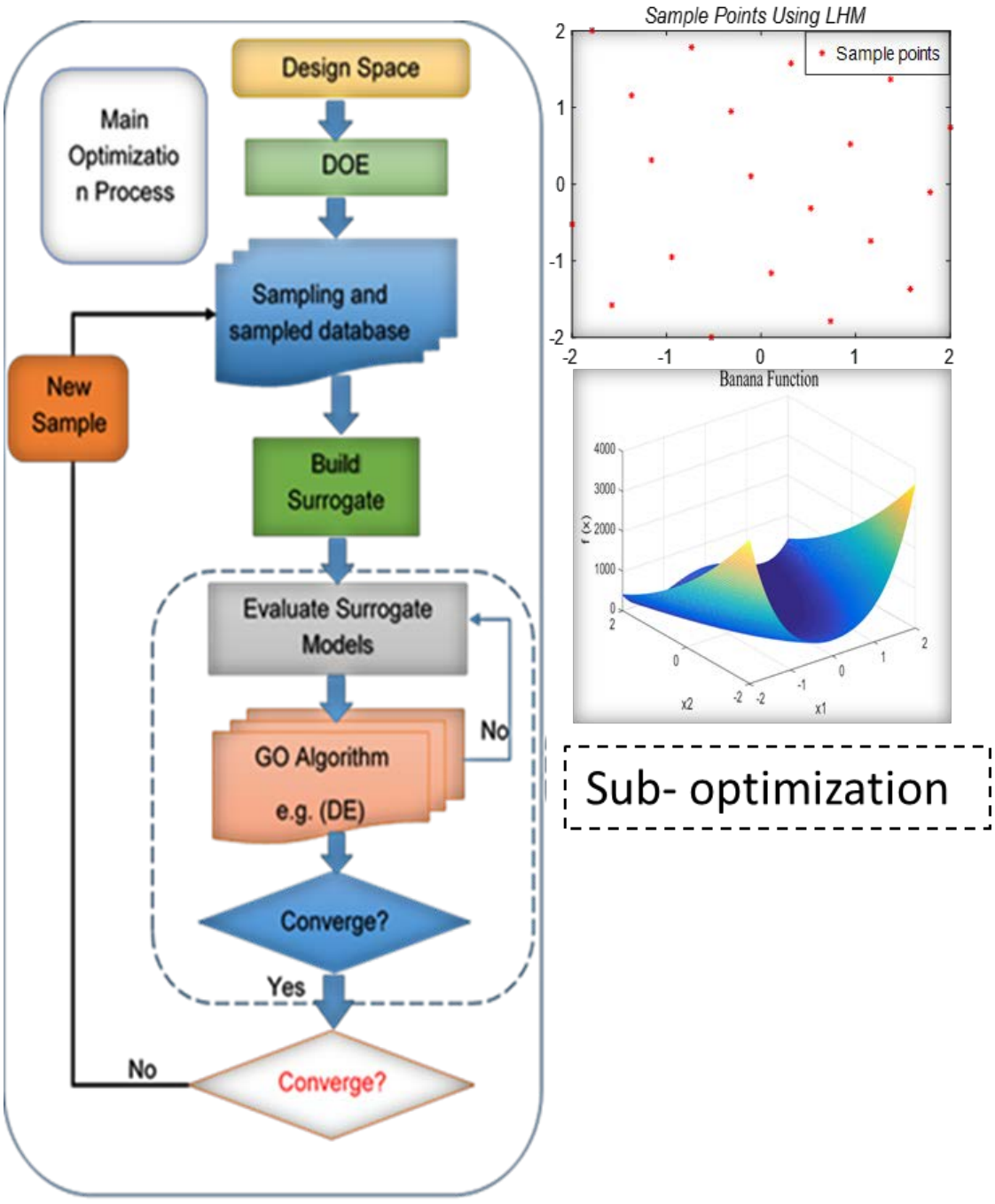


Figure 11. Flowchart of the surrogate-Assist GO Algorithms Process

2.6 Summary

In this Chapter, the background needed for this work was presented. GO approaches that have been used to tackle expensive black-box problems were reviewed. Kriging, RBF, QRF, and MARS SMs are the most commonly used metamodeling techniques. Kriging and RBF SMs are identified as the most promising and reasonable approximation models to predict the objective function of the optimization problem. Therefore, surrogate models are chosen as a means for predicting the black-box function's input variables and their intensities. It is realized that to develop a suitable SM-based optimization algorithm for expensive black-box problems, a metamodeling methodology needs to be developed first. Therefore, SM techniques are studied. The popular SM approaches were mentioned and their pros and cons were described. Then, the most suitable SM approach for expensive black-box problems was introduced. RBF and Kriging SMs were presented as approaches that have potential for being used in optimization and were described briefly.

Chapter 3. Extension of DIRECT Algorithm Using Kriging SM

3.1 Introduction

Efficient and robust non-gradient global optimization algorithms have been widely used in various engineering, science, and economics applications to overcome the limitations of conventional gradient-based optimization. One of the main drawbacks of the classical, gradient-based optimization algorithms is their inability to deal with many engineering optimization problems. This drawback occurs when the objective and constraint functions are black-box functions that are outcomes of computer simulations and the derivatives of these functions are not available.

It is very challenging to handle the black-box problems that frequently appear in practical applications using gradient-based optimization algorithms. Therefore, many non-gradient global optimization algorithms have been introduced to address this issue. Well- known non-gradient global optimization algorithms are often based on a stochastic search process, such as the GAs, PSO, ABC, and SA. These are very useful global optimization tools in solving inexpensive black-box problems. They do not require gradient information to converge to the global optimal solution, but simply use objective function evaluations to reach a solution. Another group of efficient non-gradient global optimization algorithms is based on objective-oriented sequential sampling, such as DIRECT, which describes the way the algorithm moves towards the optimum solution. The method was introduced by Jones [11] in 1993 by modifying the Lipschitzian optimization to solve nonlinear global optimization problems. As a sampling algorithm, DIRECT requires no gradient information and decides on the subsequent search based on previously collected search data. This method usually requires few function evaluations to find the area near the optimum. However, a major disadvantage of DIRECT is that the algorithm often needs many more function evaluations to find a good approximation to the global optimum solution [74]. Non-gradient global optimization methods are capable of solving complex optimization problems. However, these algorithms suffer from inefficiencies when dealing with multimodal functions and involve a high computation cost for solving EBB function. Many real-world problems in the field of engineering are EBB function-based, while most

of the algorithms in global optimization are for nonblack-box function and require a high number of function evaluation to reach a solution. Several attempts have been made trying to modify the DIRECT algorithm to improve its computational performance. Gablonsky [75] modified the original algorithm and combined it with another routine known as implicit filtering. Gablonsky and Kelley [76] introduced a form of DIRECT that is based on converging to local solution. Finkel and Kelley [77] analyzed the convergence behavior of DIRECT and proved a sub-sequential convergence result for this method. Zhu and Bogy [78] modified DIRECT to handle tolerances and to deal with hidden constraints. Huyer and Neumaier [79] implemented the idea behind DIRECT and presented a GO algorithm based on multilevel coordinate search. Chiter [80, 81] proposed a new version of potential optimal intervals for the DIRECT algorithm. Deng and Ferris [82]tried to extend this method for noisy functions and adopted a new approach that replicates multiple function evaluations per point and takes the average of the function to reduce functional uncertainty. Finally, Tavassoli *et al.* [74] improved DIRECT algorithm to deal with high dimensional optimization problems. In this Chapter, the DIRECT search algorithm, as a new capable and efficient global optimization method, has been modified using the metamodeling techniques to handle various EBB problems. This approach focuses on the accurate identification of the global optimum through a few objective function evaluations. Meta-modeling models, which are also known as surrogate models or approximation, are used to explore and provide more information on the design space that needs to be explored. Motivated by the merits of the metamodeling techniques, Kriging method is used in this work to construct the metamodeling function to guide the DIRECT search in the promising regions where the global optimum is to be identified.

3.2 The Original DIRECT Search Method

DIRECT search is one of the most well-known optimization methods used to find the global solution for certain problems without the need for the derivatives of the function in its search strategy. DIRECT derives its name from dividing the design space into a number of rectangles -one of DIRECT search key's unique features. The algorithm is a modification of the original Lipschitzian approach that eliminates the need to specify a Lipschitz constant. The main idea is to carry out intensive searches using all possible

constants from zero to infinity. The first step in the DIRECT algorithm is to subdivide the search space into three sub-hypercubes. The algorithm is then sampled at the center of each cube. Basically, DIRECT search computes the value of the function at the center-point instead of computing it at the summits. The iteration starts from the center of the design space. Subsequently, at each iteration, the DIRECT search selects and subdivides the set of hyper-cubes that are most likely to contain the lowest value of the objective function. The center point of each cube will be considered as the starting point by DIRECT, as it searches for the best value of the objective function in that specific region. DIRECT search can be used for problems where the derivatives are expensive to obtain or are unavailable. In the early years, DIRECT search algorithms were very popular because they are straightforward to implement and can be applied to many nonlinear optimization problems. In addition, DIRECT can converge to the global optimal only if the objective function is continuous or at least continuous in the neighbourhood of a global optimum. Furthermore the algorithm may quickly approach the neighborhood of global solution, but may be slow to converge to real global optimum solutions. The DIRECT deals with problems in the form of:

$$\text{Min } f(x) \tag{3.1}$$

$$\text{Subject to:} \quad x_L \leq x \leq x_U$$

where, $\boldsymbol{x_L}$ and $\boldsymbol{x_U}$ are lower and upper bounds, respectively.

3.3 The DIRECT Method Search Mechanism

The DIRECT search method solves optimization problems without demanding any information about the gradient of the optimization problem. Unlike many global optimization methods that use information about the gradient or higher derivatives to search for an optimal solution, a DIRECT search algorithm searches a set of candidate solutions around the current solution, looking for one where the value of the cost function is lower than the value at the current point. In each of the iterations, the potentially optimal hyper-rectangle is being identified and partitioned into a set of smaller spaces by trisecting it with respect to the longest coordinate it possesses. The identification of a potentially optimal hyper-rectangle would be based on its size and the value of the objective function

at its center. Thus, potentially optimal hyper-rectangles either have low function values at their centers or are large enough to be good targets for global search.

$$f(c_i) - \xi\alpha_i \leq f(\xi_j) - \xi\alpha_j, \forall j \in \mathcal{H} \tag{3.2}$$

$$f(c_i) - \xi\alpha_i \leq f_{\min} - \varepsilon|f_{\min}| \tag{3.3}$$

where α represents the size of the hyper-rectangle, calculated as the distance from the center point to the corner point of the hyper-rectangle, and assuming $\mathcal{H}$ as the index set of existing hyper-rectangles, a hyper-rectangle $i \in \mathcal{H}$ is called a potentially optimal candidate; however, there must exist a constant ξ in which $f_{\min}$ is the lowest function value available and ε is a non-sensitive value typically set as 1E-4 [83]. A graphical interpretation of DIRECT search process is illustrated in Figure 12.

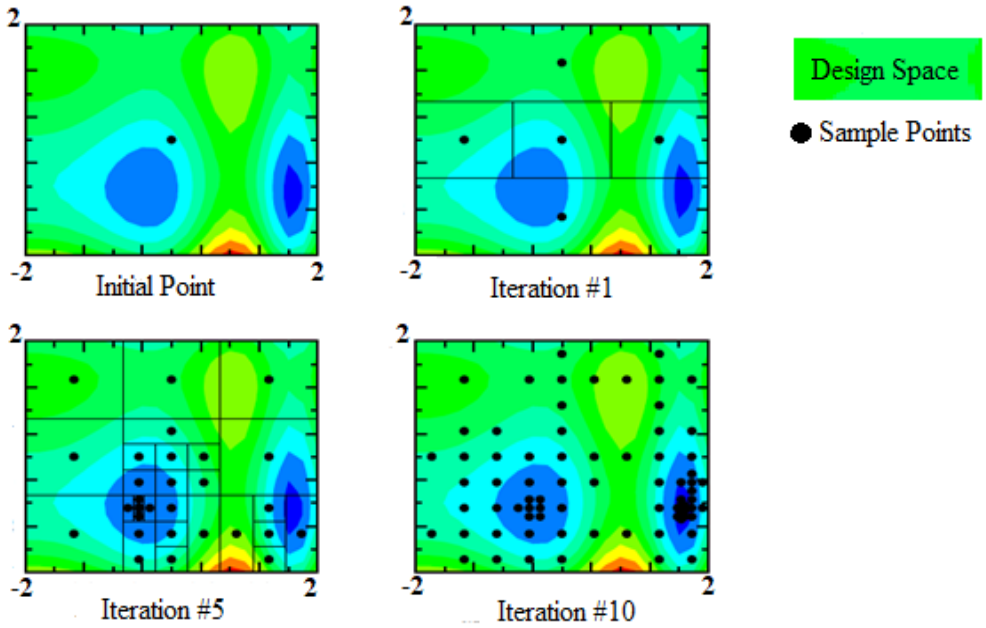


Figure 12. Searching Mechanism of the Original DIRECT Search Algorithm

Assuming an infinite number of iterations, the DIRECT search algorithm is proven to converge to the global optimum solution as long as the objective function is continuous or at least continuous in the neighbourhood of the global optimum.

3.4 Metamodeling (Surrogate) Techniques

A metamodeling (surrogate) is a "model of a model" and is an approximation to the actual model; it is often called a surrogate model. Metamodeling can also be used to create an approximate solution to the original model when very limited information about the true system is available. Metamodeling techniques are widely-used today in engineering applications to solve complex design problems due to their efficiency and flexibility. Metamodeling can predict the performance of the unknown design points through the obtained samples, and it can significantly reduce computational cost. Specifically, metamodeling play an important role in the optimization design of expensive black-box problems. There are various methods used to construct metamodeling which involve Radial Basis Functions (RBF), Kriging, Polynomial Response Surface (PRS), Support Vector Regression (SVR), and so on. RBF is expressed by a weighted sum of simple basis functions to emulate the design space. Kriging, an increasingly popular method used to predict the true design by several known samples, has some similarities with the Gaussian basis function. PRS and SVR are two kinds of regression methods commonly used to handle noisy data.

3.5 The Proposed Kriging-DIRECT Algorithm

3.5.1 Kriging Search Method

Kriging is a regression methodology that was first developed by the South African mining engineer Daniel Krige [72]. Kriging model is used widely today as a global approximation technique. Its ability to accurately interpolate response values obtained at sampling points makes it particularly attractive for approximating deterministic simulation. Previously, this method was employed to deal with gold sampling values, but now it is widely applied in the engineering design. Kriging can perform well on non-linear and multimodal problems. The modeling process is summarized as follows:

Assume that there are n sample points and evaluate their function values.

$$\boldsymbol{x} = \{x_1, x_2, \dots, x_n\}, \qquad x_i \in \mathbf{R}^m$$

$$\mathrm{Y} = \{y_1(x_1), y_2(x_2), \dots, y_n(x_n)\} \tag{3.4}$$

where *m* is the dimension of sample points. The predictor can be expressed as follows:

$$\hat{y}(x) = f^T(x)\hat{\beta} + \boldsymbol{r}^T(x)\,\hat{a} \tag{3.5}$$

where parameters $\hat{\beta}, \hat{a}$ can be obtained by likelihood estimation. Here $\boldsymbol{r}^T(\boldsymbol{x})$ is a vector which represents the correlation between a to-be-tested point and all the known samples.

$$\boldsymbol{r}^T(\boldsymbol{x}) = \{R(\boldsymbol{x}, \boldsymbol{x_1}), R(\boldsymbol{x}, \boldsymbol{x}_2), \ldots, R(\boldsymbol{x}, \boldsymbol{x}_n)\}$$

$$\hat{\beta} = (\boldsymbol{F}^T \boldsymbol{R}^{-1} \boldsymbol{F})^{-1}\, \boldsymbol{F}^T\, \boldsymbol{R}^{-1}\, \boldsymbol{Y} \tag{3.6}$$

σ^2 = 1/n (Y- F β^) R^(-1) 〖(Y-Fβ^)〗 ^T

From Figure 13, it can be observed that the Kriging model has a good approximation performance; thus, the prediction model of Kriging can be more accurate by increasing the number of sample points in the design space.

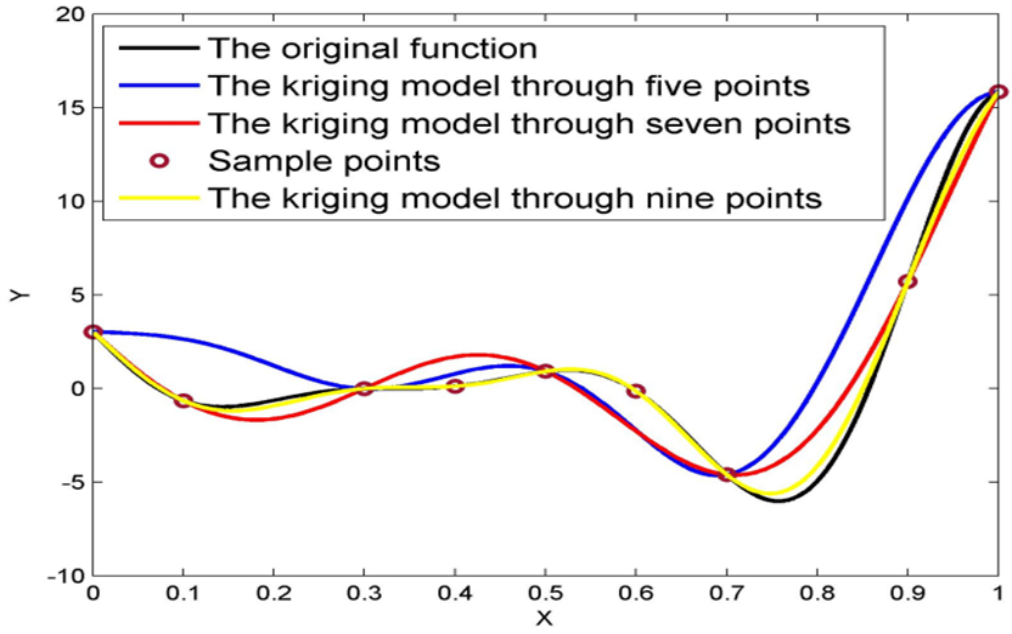


Figure 13. One Dimensional Function Prediction Using Kriging

In Equation (3.6), F consists of $f(\boldsymbol{x})$ which is the basis function of $\boldsymbol{x}$. $\boldsymbol{R}$ is the correlation matrix.

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}(\boldsymbol{x_1}, \boldsymbol{x}_2) & \cdots & \boldsymbol{R}(\boldsymbol{x}_1, \boldsymbol{x_n}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{R}(\boldsymbol{x}_n, \boldsymbol{x_1}) & \cdots & \boldsymbol{R}(\boldsymbol{x_n}, \boldsymbol{x_n}) \end{pmatrix} \tag{3.7}$$

$$R(x_i, x_j) = \exp\left(-\sum_{k=1}^{m} \theta_k \left| \boldsymbol{x}_i^k - \boldsymbol{x}_j^k \right|^2\right) \quad (3.8)$$

3.5.2 DIRECT Algorithm integrated with Kriging metamodeling

The DIRECT search methods remain popular because of their simplicity, flexibility, and reliability. However, the DIRECT search places a high emphasis on global search, which leads to slow convergence to the actual global optimum. This work proposes a new strategy that can help the DIRECT search algorithm to reach the optimum region with a fewer number of function evaluations. In this research, we are interested in cases where the function evaluation is costly and obtaining an approximate solution is the goal.

3.5.3 The Proposed Kriging-DIRECT Algorithm

The basic steps of the new proposed algorithm are shown in Figure 14 and outlined as follows:

Step 1: Run DIRECT for several iterations and evaluate a set of expensive points, x_i, $i = 1, \ldots, n$ over the design space;

Step 2: Build a surrogate model using Kriging SM

Step 3: Add more cheap design points over the Kriging SM using HLD and evaluate those points to choose the best one.

Step 4: Use the better objective value as new initial point for DIRECT Search and identify the new upper and lower boundaries of the promising region.

Step 5: Run the DIRECT Search using the better point from the model as reference (center point)

Step 7: Repeat until the stopping criteria has been met.

3.5.4 Optimization process using the Kriging-DIRECT algorithm

The introduced algorithm is a combination of the DIRECT search with metamodeling in order to reduce the high number of function evaluations required by the DIRECT search to reach global solution. The Integration of the DIRECT search with metamodeling will assist the DIRECT search algorithm to converge quickly to the global solution with reasonable number of function evaluations. K-DIRECT algorithm can effectively improve the accuracy so that the optimal value obtained is much closer to the theoretical one when

compared with the standard DIRECT algorithm. The new methodology significantly will improve the convergence rate of the DIRECT search algorithm. As described in the Chapter Two, using Kriging technique in optimization requires fitting the kriging model, finding the points that maximizes expected improvement and evaluating the objective function with low cost. The first step of this optimization procedure is to run the DIRECT search for four iterations. The second step is built the Kriging using the expensive points evaluated by the DIRECT search. Once the Kriging model is built using a set of training data from step two, the Kriging model have to be evaluated to obtain the best value of the objective function. After finding an optimum design by the Kriging model, this design evaluation points has to be added to the training data set. The DIRECT search then will use the better point as initial point to find the global solution. Then the Kriging model has to be rebuilt and again research the surrogate model. This process is iterated until we reach some convergence criterion. The performance of the proposed algorithm is tested on computationally expensive single objective numerical optimization, with different mathematical properties.The comparison of the two methods with convergent curves is shown in the next section.

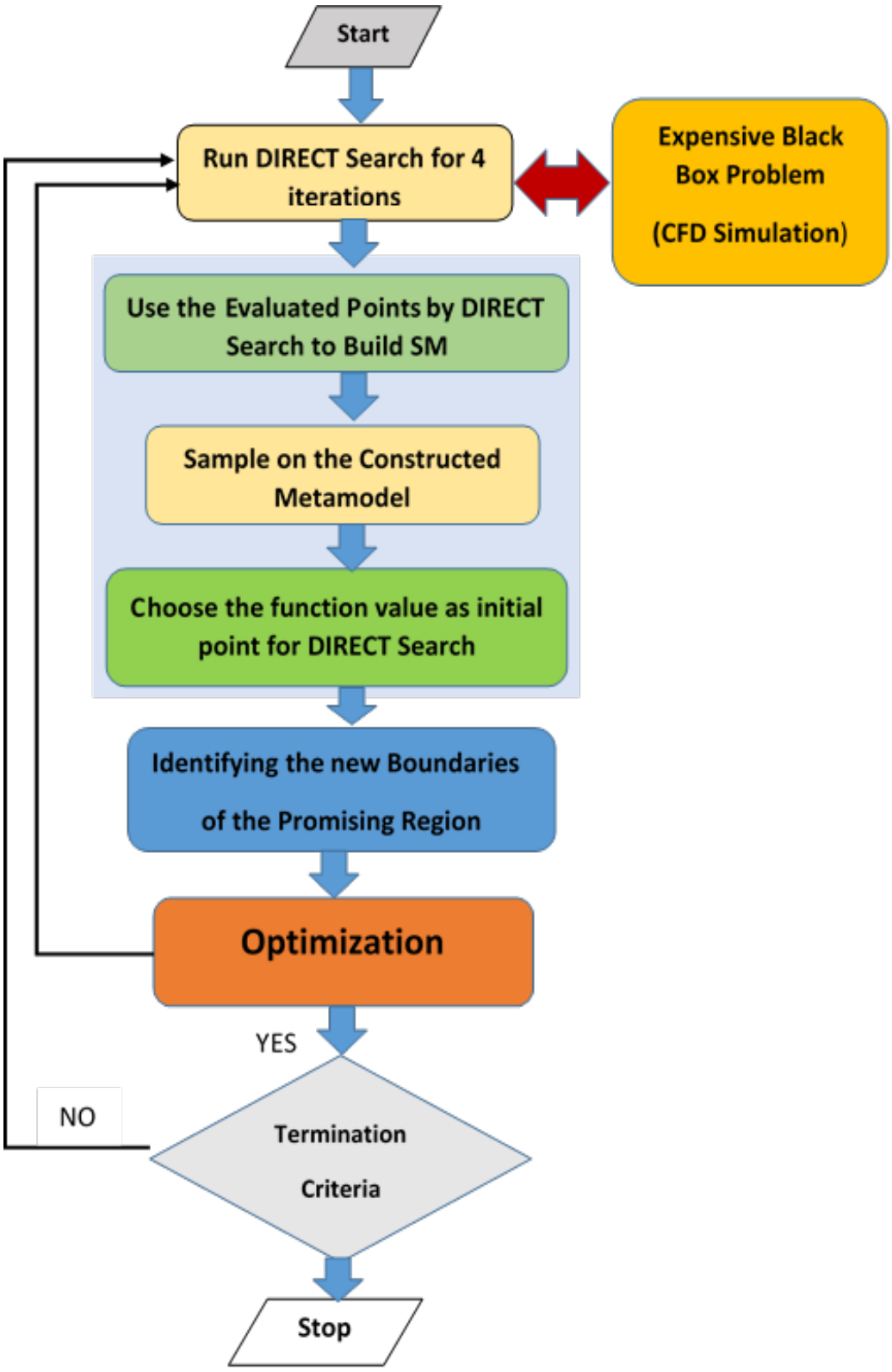


Figure 14 Flowchart of Kriging-DIRECT Algorithm

3.6 Testing of the Kriging-DIRECT algorithm

In this section, to investigate the effectiveness of the proposed Kriging-DIRECT algorithm, we conducted an empirical study on ten widely used benchmark problems. Every benchmark function has its own properties whether it is unimodal, multimodal, separable or non-separable. It is noteworthy that the combination of these properties determines the complexity of the functions. A function is considered multimodal if it has two or more local optima, and it is considered separable if it can be rewritten as a sum of function just from one variable. The characteristics of these test problems are listed in Table 2. These included ten problems (Banana, Zakhaov, SC, Beale, Peak, Shub, Dixion and Price, Perm, Leon, Easom) [84]. The aim of the optimization process is to obtain the global optima; therefore, the regions around local optima should be avoided because the DIRECT search algorithm might get stuck in local optima and consider that local optima as the global optima. Another important property that determines the difficulty of the problem is the dimension of the search area.

Table 2: Benchmark Function selected for Validations

No.	Fun	D	Region	f*	Fun. Properties
1	*Banana*	*2*	$[-2, 2]^2$	*0.0000*	*U-Modal*
2	*Zakharov*	*2*	$[-5, 10]^2$	*0.0000*	*U-Modal*
3	*SC*	*2*	$[-2, 2]^2$	-10.3223	*M-Modal*
4	*Beale*	*2*	$[-4.5, 4.5]^2$	*0.0000*	*M-Modal*
5	*Peaks*	*2*	$[-3, 4]^2$	*0.0000*	*M-Modal*
6	*Shubert*	*2*	$[-10, 10]^2$	-186.7309	*M-Modal*
7	*Dixon and Price*	*3*	$[-10, 10]^3$	*0.0000*	*M-Modal*
8	*Perm*	*3*	$[-2 , 2]^3$	*0.0000*	*U-Modal*
9	*Leon*	*4*	$[-1.2, 1.2]^4$	*0.0000*	*U-Modal*
10	*Easom*	*4*	$[-10, 10]^4$	*0.0000*	*M-Modal*

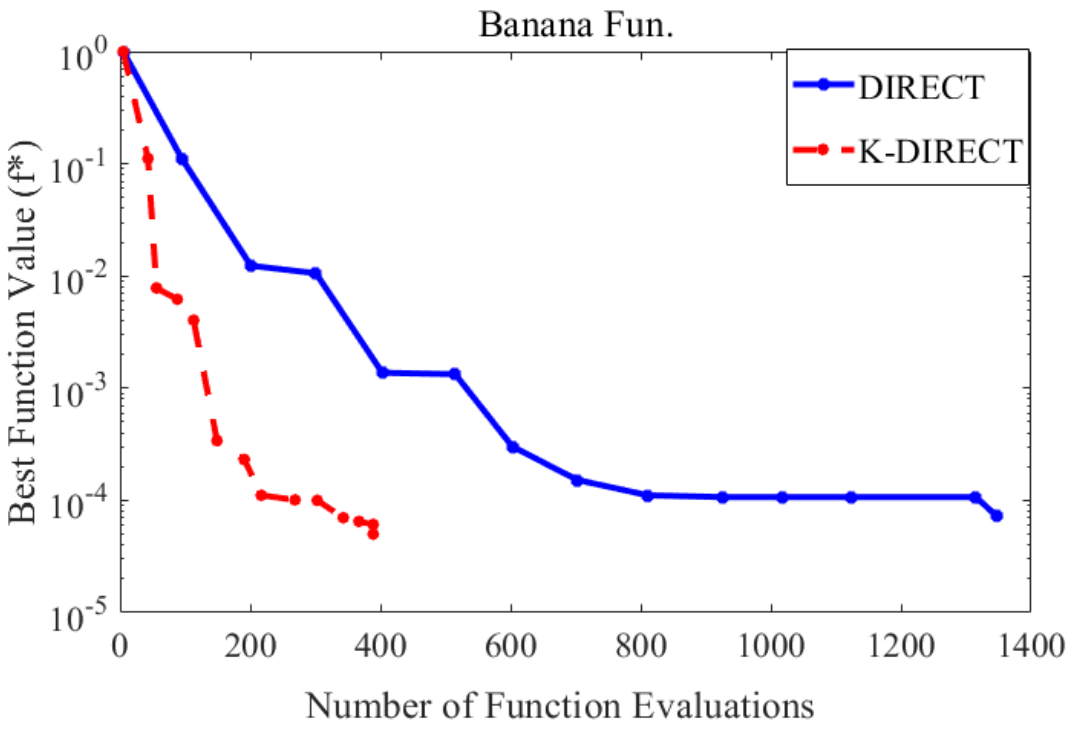


Figure 15. Test Problem #1.

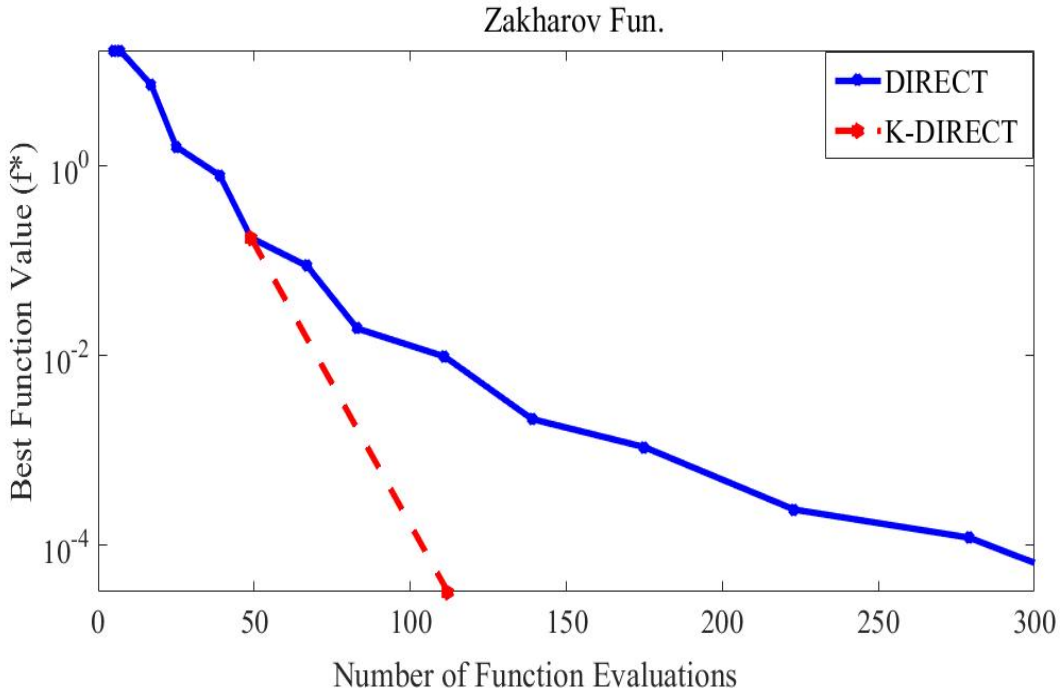


Figure 16. Test Problem #2.

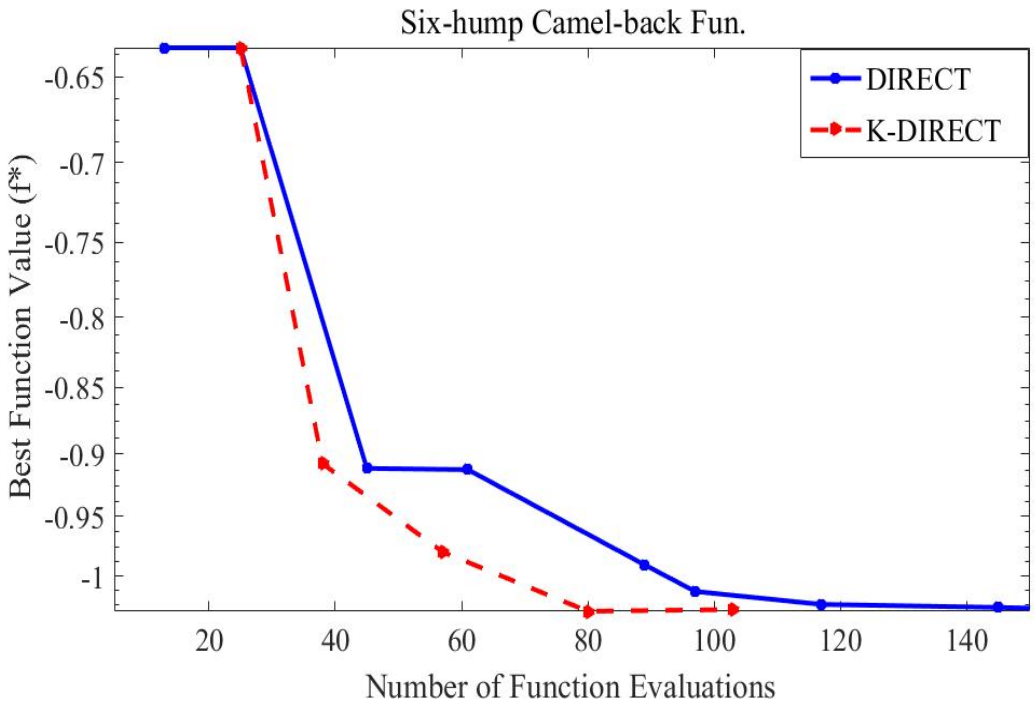


Figure 17. Test Problem #3.

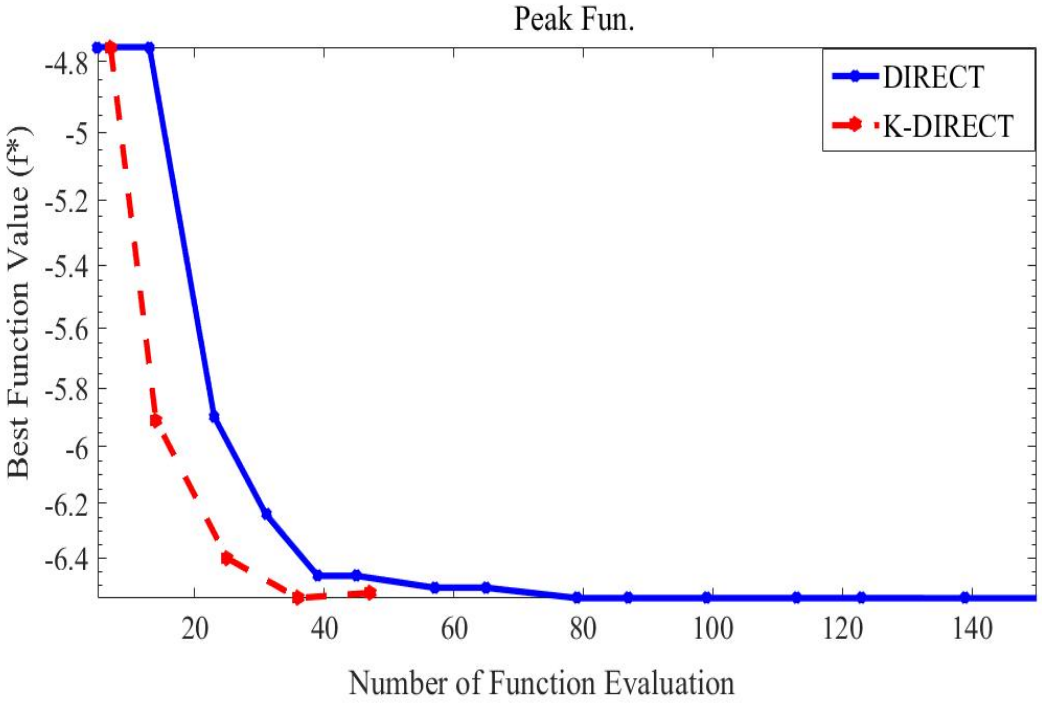


Figure 18. Test Problem #5.

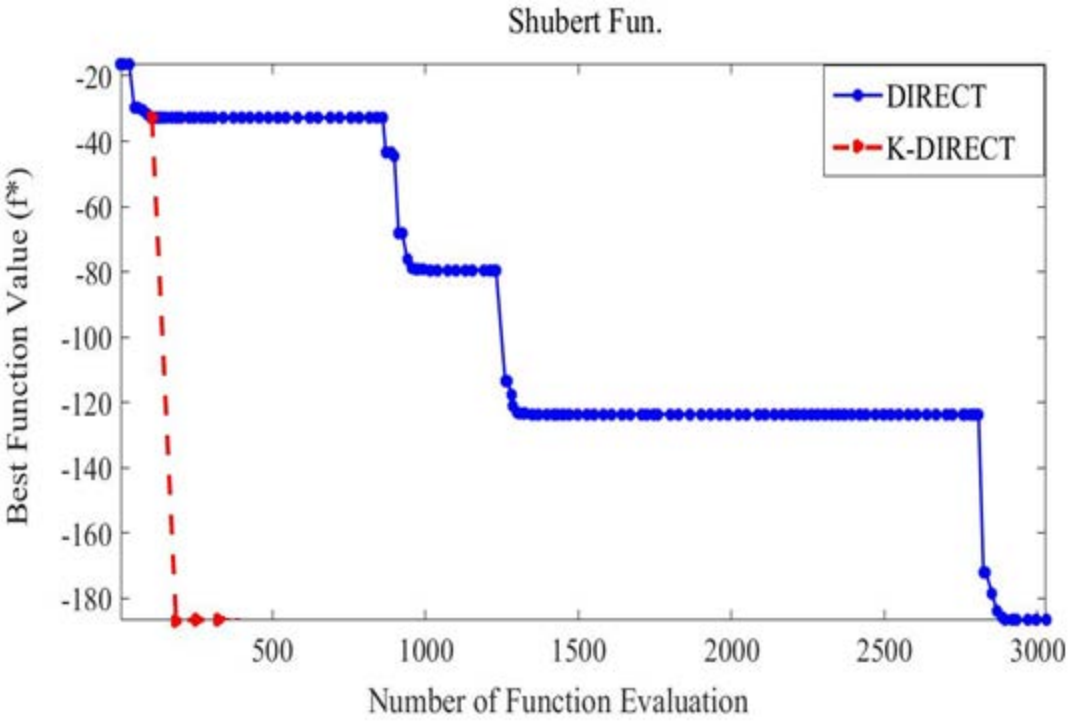


Figure 19. Test Problem #6.

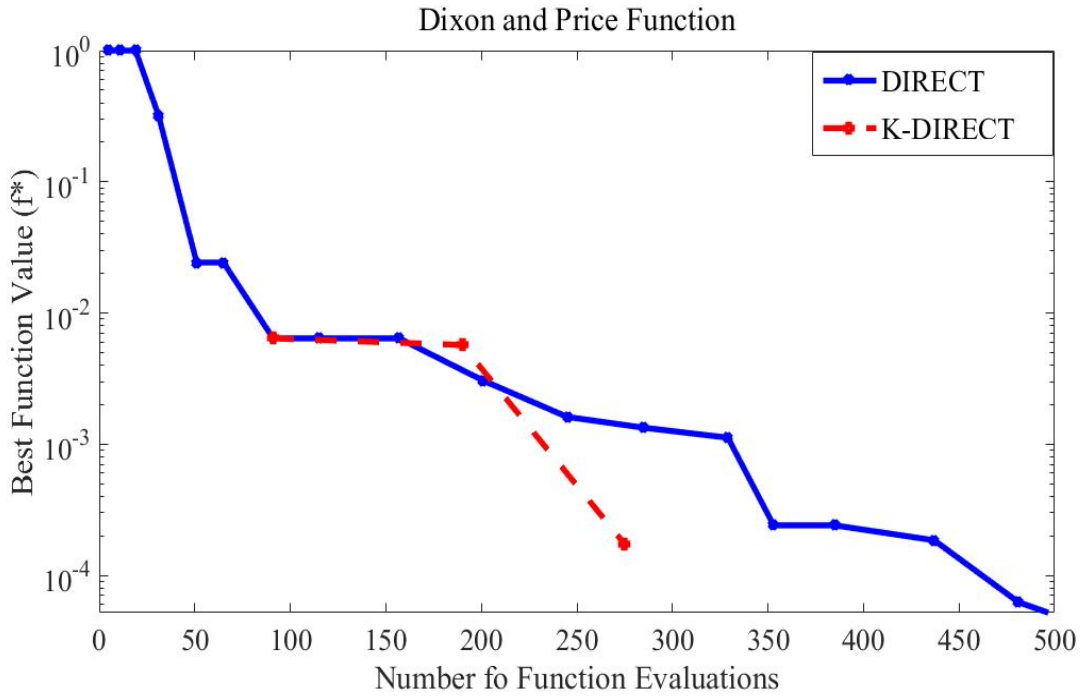


Figure 20. Test Problem #7.

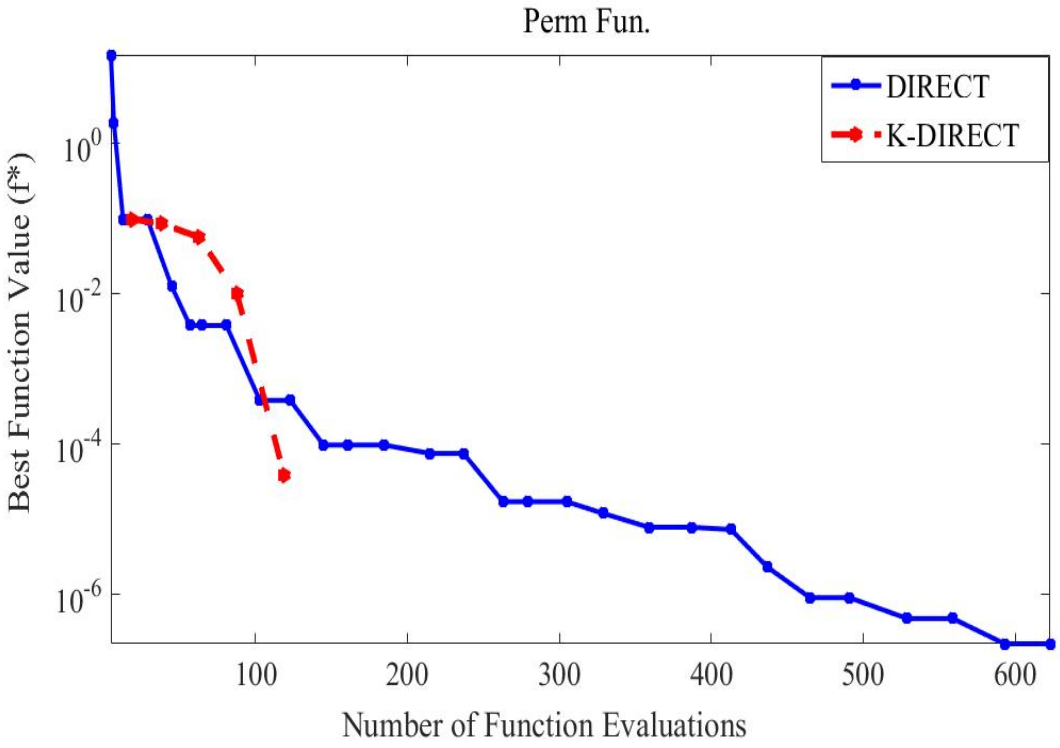


Figure 21. Test Problem #8.

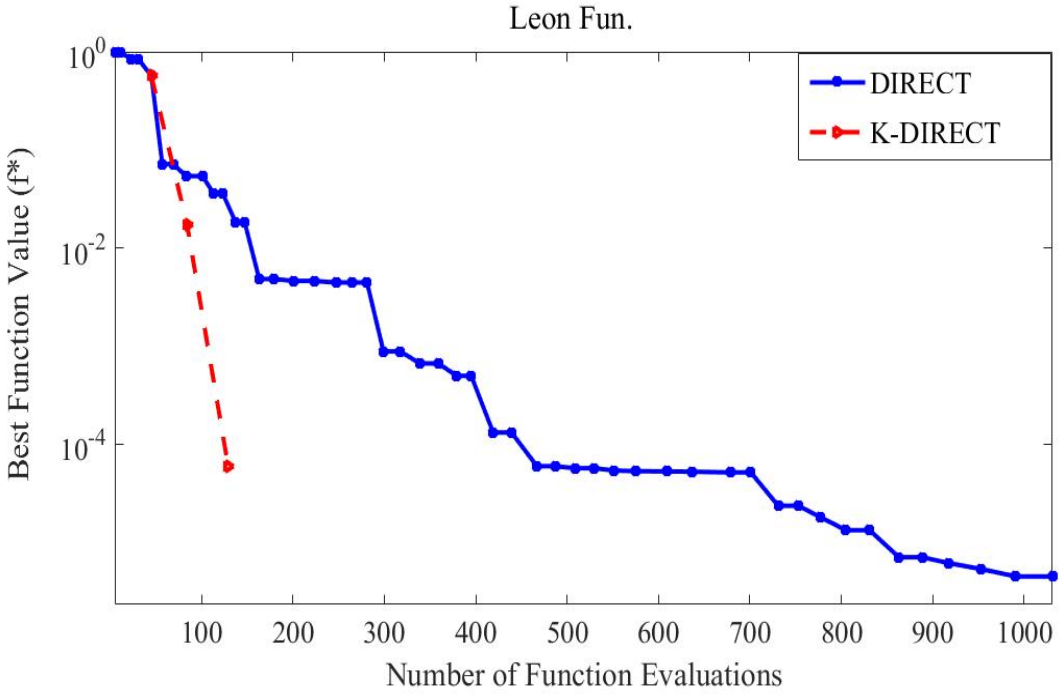


Figure 22. Test Problem #9.

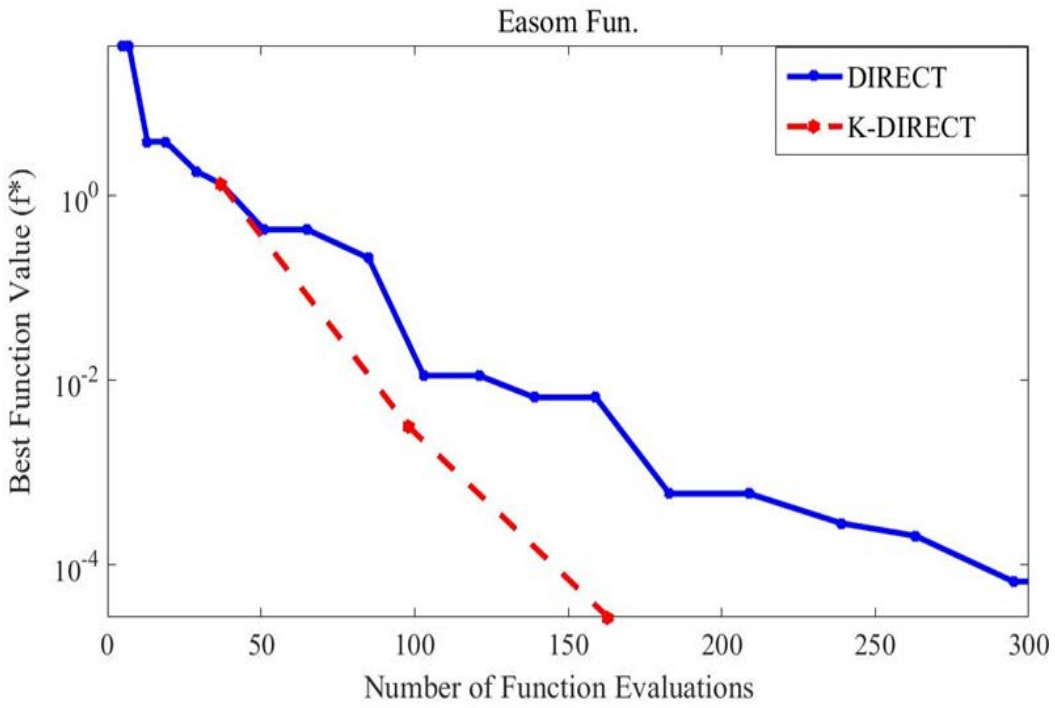


Figure 23. Test Problem #10.

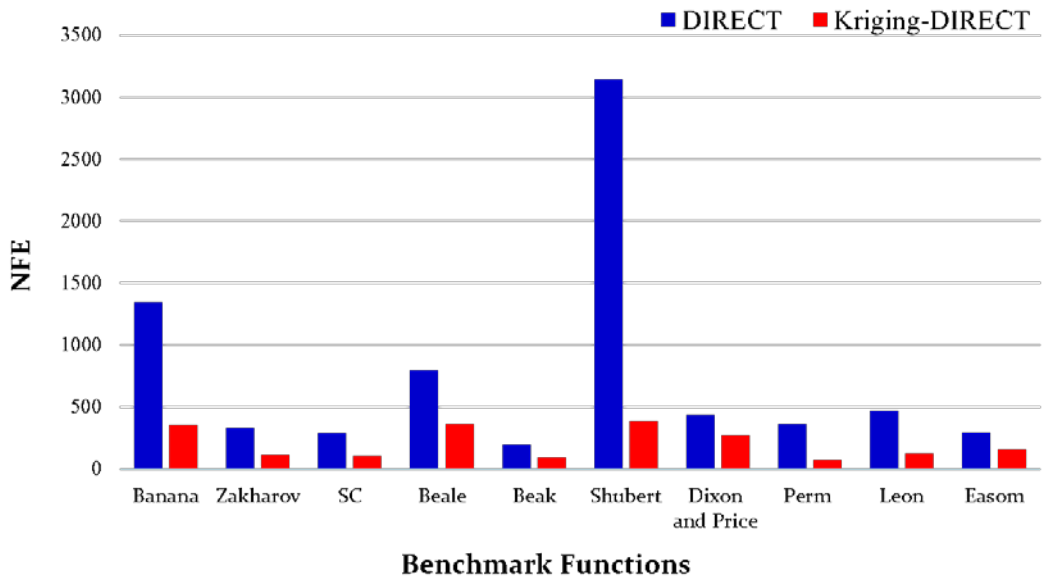


Figure 24. NFE needed by DIRECT versus Kriging-DIRECT

Table 3: Comparison Results of Kriging-DIRECT Vs DIRECT Search

Fun	Search Space	DIRECT		Kriging -DIRECT		N.F.E Saved	
		Min	NFE	Min	NFE	NFE	%
f1	*[-2 2]*	*7.2 E-5*	*1397*	*2.033 E-5*	*398*	*999*	*71 %*
f2	*[-5 10]*	*2.646 E-5*	*331*	*3.242 E-5*	*112*	*219*	*66 %*
f3	*[-2 2]*	*-1.0316*	*289*	*-1.0316*	*104*	*185*	*64 %*
f4	*[-4.5 4.5]*	*4.0 E-5*	*793*	*3.0 E-5*	*362*	*431*	*54 %*
f5	*[-3 3]*	*-6.5511*	*195*	*-6.5511*	*89*	*106*	*54 %*
f6	*[-10 10]*	*-186.730*	*3143*	*-186.7303*	*391*	*2752*	*87 %*
f7	*[-10 10]*	*1.849 E-4*	*437*	*1.7287 E-4*	*275*	*162*	*37 %*
f8	*[-2 2]*	*7.888 E-6*	*359*	*5.611 E-6*	*73*	*286*	*80 %*
f9	*[-1.2 1.2]*	*5.968 E-5*	*467*	*6.002 E-5*	*129*	*338*	*72 %*
f10	*[-10 10]*	*6.5835 E-5*	*295*	*2.6951 E-5*	*163*	*132*	*45 %*

3.7 Comparisons and Discussion

In this section, the results obtained from the new Kriging-DIRECT algorithm are compared with those of the original DIRECT method, using ten benchmark problems that have different levels of complexity. For a fair comparison, the functions are selected with different shapes including unimodal and multimodal with many local minima and plate-shaped functions, which are shown in Table 2. The algorithm was coded using Matlab R2014b, and was run on a PC with a 3.4 GHz Core I7 processor with 16 GB RAM, and windows 10. Table 3 provides computation results obtained from commonly used benchmark test problems. Every benchmark function is optimized using Kriging-DIRECT and DIRECT search algorithm. Ten independent runs were made, and the convergence trend of these problems are ploted in Figure 15 to Figure 23. These results are related to the functions 1 to 10 respectively. The number of function evaluations needed by Kriging-DIRECT algorithm are much fewer than what DIRECT search needs to reach the same results. For instance, multimodal test functions including SC, Shubert and Peaks, Kriging-DIRECT algorithm requires 104, 89 and 391 function evaluations to reach the satisfactory solution, while for the same test function, the DIRECT search algorithm needs 289, 195 and 3,143 function evaluations, respectively, to reach almost the same solution. That clearly reflects the computational efficiency and capability of the new proposed Kriging-DIRECT algorithm.

In Table 3, the DIRECT search algorithm is considered to be a reference when computing the relative number of function evaluations. Banana, Perm, and Shubert benchmark test

problems are selected as examples to show the convergence trends of both algorithms. A comparison of both optimization algorithms on the tested benchmark problems is graphically summarized from Figure 15 to Figure 23 . Also a comparison of the number of function evaluations required by DIRECT and Kriging-DIRECT algorithms, for all tested problems, is shown in Figure 24. The result also indicates that the performance of the Kriging-DIRECT algorithm is better than the DIRECT search on a selected set of benchmark problems. The significant improved performance of Kriging-DIRECT over DIRECT can be observed in the tested problems including Banana, Easom, and Dixon and Price. From the convergence plots provided, it can be seen that the convergence speed of Kriging-DIRECT algorithm is better than the DIRECT search on the entire set considered. The study concludes that Kriging-DIRECT algorithm requires a significantly fewer number of function evaluations to converge to the same accuracy than needed for the original DIRECT search for all chosen problems. For a better illustration of the effects of using a surrogate model with the DIRECT search, a search performance metamodeling is a unique technique as it guides the DIRECT search to achieve high performance with the lowest number of function evaluations. The proposed approach effectively moves to attractive regions and converges rapidly to the global minima. Also from the above convergence history of the tested functions, note that the presented algorithm is capable of handling and solving the challenging benchmark in expensive black-box (EBB) problems form; it locates the optimum with comparable accuracy and obtains the results with a much reduced number of objective function evaluations and computation time Figure 15. Concerning the functionality of the method when used with other cases, however, more tests may be needed. While improving the capacity to handle high-dimensional problems and complex cases is still undergoing research, continuing work in this research direction involves testing the method on high-dimensional and varied real-life applications and other cases.

3.8 Advantages of the Proposed Strategy

The Kriging-DIRECT method presented above offers a number of advantages when compared to the original DIRECT search algorithm. The most important advantage is that the new method includes:

- The proposed algorithm does not use a high number of function evaluations to converge
- Quickly starts converging to the promising region where the global optimum is located
- The approximation accuracy and the performance of the Kriging SM has very high accuracy and effective approximation to the actual function.
- The algorithm is capable of solving the challenging benchmark problems in global optimization; locating the optimum with comparable accuracy; and obtaining the results with much reduced NFE.

3.9 Summary

This Chapter presents contributions that have been made to extend the DIRECT search algorithm to enable it to be used for expensive black-box problems. As mentioned in Chapter Two, Kriging is identified as a powerful metamodeling techniques and works very well for low and medium size optimization problems. DIRECT search algorithm normally requires extensive computation for complex, black-box engineering problems. This work introduces an extension of the conventional DIRECT search algorithm by adding a Kriging metamodeling scheme to improve the search efficiency. The new approach utilizes the Kriging metamodeling technique to construct an approximation model of the optimization problem to guide and constrain the DIRECT algorithm to search in the promising region only. Ten representative local and global optimization benchmark problems have been used to test the effectiveness and efficiency of the newly introduced algorithm. The test results showed considerable improvement in terms of convergence rate, quality of solution, and robustness. In most cases, the number of function evaluations required by the new Kriging-DIRECT method were reduced by more than 50% due to the avoidance of wasteful searches over local minima. In the future, a mixture of multiple metamodeling techniques may be used to further improve the efficiency of this new method for high-dimensional and computation intensive problems.

Chapter 4. Extension of DE Algorithm Using RBF SM

4.1 Introduction

DE, which is a heuristic and nature-inspired optimization algorithm, has been introduced by Storn and Price [85]. It has attracted much attention from scientists to be used for solving engineering and non-engineering complex optimization problems. Generally, the DE method is considered to be a fast, efficient, and robust procedure, which represents an ideal alternative to EA. Over the last two decades, the DE has been effectively applied to different kinds of applications such as optimal design of shell-and-tube heat exchangers [85], generation planning problems [86], capacitor placement problems [87], chaotic systems control and synchronization [88], and induction motor identification problems [89]. The objective of this study is to find the global optimum solution of the optimization problem.

An optimization problem is considered to be expensive black-box as long as its computation time is high and the algorithm takes a long time to search and converge to a global solution; therefore, the classical DE cannot be applied to solve computationally expensive black-box problems unless a modification is added. In today's world, many computationally intensive and expensive mathematical or physical models and engineering problems exist. To solve these computationally expensive problems, an enormous number of fitness function evaluations are required during the evolution process when evolutionary algorithms (EAs) are used. To be able to solve these problems, it is crucial to carefully explore and search the design space. Recently, EA-based surrogate models have attracted much attention by researchers due to their efficiency. In such algorithms, surrogate model, also known as meta-modeling, is used to evaluate the objective function by constructing an approximate model [90]. The use of surrogate models with GO algorithms has largely contributed towards reducing the computations required to converge to the global solutions in computationally intensive engineering design problems. The Surrogate model replaces the expensive black-box functions with cheap, easy to construct, and visible functions that do not require powerful personal computers (PCs) and do not take much time to evaluate. Global optimization using approximated or surrogate models as key to reduce design space has attracted considerable interest lately due to its high efficiency, robustness, and ease of

implementation. In this Chapter, we aim to develop a new algorithm based on RBF SM with the DE algorithm that is able to obtain the global solutions to medium scale expensive optimization problems with limited budget of expensive function evaluations. Combining and utilizing the DE with the RBF SM will increase the convergence speed, hence reducing the number of expensive function and constraint evaluations.

4.2 Brief literature Overview the DE

Differential Evaluation (DE) is a stochastic population-based search method for solving non-linear, high-dimensional, and complex computational optimization problems. The DE is considered the most recent EA for solving real-life optimization problems. The DE has many advantages including simplicity of implementation, reliability, robustness, and, in general, is considered as an effective global optimization algorithm [91]. Therefore, the DE has been applied in solving many real-world problems, such as pattern recognition studies [92], financial markets dynamic modeling [93], signal processing problems, power systems [94], and many others. Due to some drawbacks of the DE – such as weak global exploration ability and sensitivity to the choice of the control parameters – many researchers have proposed serious modifications to improve the overall performance of the DE algorithm. A Fuzzy Adaptive Differential Evolution (FADE) algorithm was proposed by Liu and Lampinen [95]. Brest *et al.* [96] proposed a new version of DE for self-adapting control parameter settings that showed unique performance on numerical benchmark problems. . Ali and Torn [97] proposed a new DE algorithm with two evolving populations. Zaharie [98] introduced an adaptive DE (ADE) algorithm based on the idea of controlling the population diversity and implemented a multi-population method. SaDE (self-adaptive differential evolution) is proposed by Qin *et al*. [99] .The main idea of SaDE is to simultaneously implement two mutation schemes: “DE/rand/1/bin” and “DE/best/2/bin” and also to adapt mutation and crossover parameters. Omran et al [100] proposed a Self-adaptive Differential Evolution (SDE) algorithm. The scaling factor F is self-adapted using a mutation rule similar to the mutation operator in the basic DE. Zhang and Sanderson [101] introduced a new Differential Evolution (DE) algorithm, named JADE, to improve optimization performance by implementing a new mutation strategy ‘‘DE/current-to-best’’ with optional external archive and updating control parameters in an adaptive manner. The

scale factor of local search differential evolution (SFLSDE) is presented by Neri and Tirronen [102]. Many other modifications to the DE have been developed in order to improve the global performance of basic the DE, to enhance the local exploitation tendency, and to improve the convergence rate of the DE algorithm.

4.3 The Differential Evaluation Algorithm (DE)

Differential Evaluation algorithm is a well-known evolutionary algorithm. The DE is one of the most powerful population-based search global optimization algorithms and can be considered similar to GA since it utilizes similar operators: crossover, mutation, and selection. The key difference between the GA and the DE is in the creation of better solutions, where DE depends on mutation operation and GA depends on crossover operation. Since DE depends on mutation operation, it employs the mutation as a search mechanism and takes advantage of the selection operation in order to direct the search towards the promising regions in the design space. Target Vector, Mutant Vector, and Trail Vector are three properties that the DE uses for creating a new population size. The target vector is the vector that contains the solution for the search space; the mutant vector is the mutation of the target vector; and the trail vector is the resultant vector after the crossover operation between target vector and mutant vector. As stated before, the basic steps of the DE algorithm are similar to GA with only slight differences. The DE starts with steps such as population initialization followed by evaluation to determine the fittest members of the population. Later, new parameter vectors get generated by adding the weighted difference of the two population vectors with the third vector. This step is referred to as mutation. Within the crossover, the vector is mixed and the algorithm takes a final step of selection. In order to see the differences between the DE and the GA, a more detailed discussion on the three main operators in the DE is required.

In the mutation step, each of the *N* parameter vectors goes through mutation. Mutation is the operation of expanding the search space, and a mutant vector is generated by:

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \tag{4.1}$$

Where, *F* is the scaling factor with a value in the range of [0, 1] with solution vectors $\boldsymbol{x_{r1}}$, $\boldsymbol{x_{r2}}$, , and $\boldsymbol{x_{r3}}$, being chosen randomly and satisfying the following criteria:

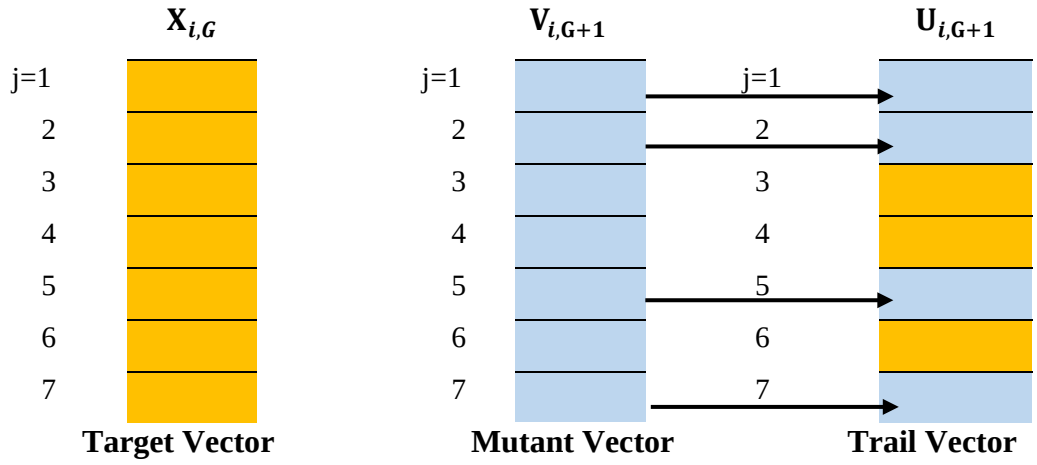


Figure 25. Illustration of DE Crossover Process with vector dimension of 7

$$x_{r1}\,,x_{r2}\,,x_{r3}\,,\quad,\quad r_1 \neq r_2 \neq r_3 \neq i \tag{4.2}$$

Here, *i* is the index of the current solution. Figure 25 illustrates a two-dimensional vector which plays a part in generating the mutant vector. The crossover operation is introduced to increase the diversity of the disconcerted parameter vectors. The parent vector is mixed with a mutated vector and a trial vector is produced by:

$$u_{i,G+1} = \begin{cases} v_{i,G+1}\ if\ R_i \leq CR \\ x_{i,G}\quad if\ R_i \leq CR \end{cases} \tag{4.3}$$

Here, CR is a crossover constant and *Rj* is a random real number between [0,1] with *j* denoting the *jth* component of the resultant array. In the DE algorithm, all candidate solutions in the population have the same chance of being selected as parents without considering their fitness value. This is the main difference in the operations of the DE and the GA. That is, the child (trail vector) produced is only evaluated after mutation and crossover operations. After that, the performance of this child vector is compared to its parent and the better vector is retained in the population. The exploitation behaviour occurs when the difference between two solution vectors in Eq. 4.1 are small, while the exploration behaviour occurs when the difference between the two is large. The DE is advantageous in terms of enhancing the capacity of local search and keeping the multiplicity of the population while it suffers from slow convergence.

4.4 Surrogate Model (Metamodeling) Techniques

4.4.1 Overview of Surrogate Models

Approximation models or surrogate models (Metamodeling) play a major role in metamodeling-based global design optimization. The surrogate in a simple, easy-to-calculate form, is used to replace the original, black-box computer analysis and simulation model. The introduced metamodeling also provides insight into the optimization problem by visualizing the interactions among design variables, objective functions, and constraints. The overall objective is to reduce the computation cost of computationally intensive design simulations and analyses, using inexpensive surrogates of these analyses and simulations [4]. The main benefits of surrogate models can be summarized as follows:

- It is much cheaper to evaluate surrogate model than to perform a complex computer simulation. This yields a reduction in computational effort where many function evaluations are necessary (e.g. in optimization or stochastic analyses).
- By the use of surrogate models, the designer can easily explore the entire design space to get a more profound understanding of the system under investigation.
- Surrogate models can be used to combine information gathered from different sources, for instance analysis codes for different disciplines (e.g. fluids, structures, or thermodynamic problems), or physical experiments and computer simulations.
- Parallel computing is simple since, in general, the individual sampling points are appointed simultaneously. Hence, the necessary computer experiments can be performed independently and in parallel.
- Surrogate models can be used to smooth response values if noise is present in the observations.
- In the next subsection, RBF model, which is used in this work, is introduced.

4.4.2 Radial Basis function

Originally introduced by Hardy [69] and further improved by Dyn [103], Radial Basis Function (RBF) is an effective algorithm for smoothing and interpolating experimental data. The form of the approximate model is a basis function of the Euclidean distance between the sampled data point and the point to be predicted. Developed as an analytical method for representing irregular surfaces, the RBF uses linear combinations of radial

symmetric functions of the Euclidean distance to build approximation models. The model can mathematically be expressed by the following equation:

$$\hat{y}(x) = \sum_{i=1}^{N} w_i \, \varphi(\|x - c_i\|) \tag{4.4}$$

Where, the approximated function $\hat{y}(x)$ is represented as a sum of N radial basis functions φ, each associated with a difference center c_i and weighted by an appropriate coefficient, w_i RBF approximation is capable of producing good fits to arbitrary contours of both deterministic and stochastic response functions. The radial function $\varphi(r)$ can take many forms as shown in Table 4.

Table 4: RBF Forms

Function Type	Radial Basis Form
Linear Function	$\varphi(r) = r$
Cubic Function	$\varphi(r) = r^3$
Thin Plate Function	$\varphi(r) = r^2 \log r$
Multi-quadratic Function	$\varphi(r) = \begin{cases} \sqrt{r^2 + a^2}, & r > 0 \\ 0, & r > 0 \end{cases}$
Gaussian Function	$\varphi(r) = e^{-ar^2}$

The RBF passes all the expensive points obtained using the original numerical analyses/simulations, prevents unnecessary curvature added to the unknown function, and preserves the minimum from these expensive points. The RBFs are simple, intuitive, and easy to construct. As a result, they have been successfully used in many engineering applications including ocean depth measurement, altitude measurement, surveying, mapping, geography and geology, and medical imaging.

4.5 The Proposed RBF-DE Algorithm

There has been much attention paid to hybrid optimization algorithms in the past few decades, and more attention is given to the optimization algorithm based meta-modeling techniques. Recently, researchers have been interested in combining meta-models (either Kriging, polynomial, RBF, etc.) with evolutionary algorithms (EA), which is one of the categories of nature-inspired global optimization approaches when dealing with the different types of optimization problems. Over the years, considerable progress has been achieved in developing more flexible, capable, and efficient nature-based global

optimization methods. Nature-based algorithms are based on random observations that give different final solutions each run, starting from an identical initial point. Specifically, EA population-based approaches deal with a set of candidate solutions that can be improved via a number of iterations. Evolutionary algorithms are the optimization algorithms based on Darwin's principle which uses four general steps: reproduction, crossover, mutation, and selection. Finally, the fitness function is used to reach the optimal solution. In this work, the RBF model is utilized to assist the DE in the search for finding the global solution of expensive black-box problems with reasonable cost. The integration of the DE and the RBF is unique in terms of the search capability and the speed of convergence.

4.5.1 Steps of the proposed algorithm

In the new proposed algorithm, a number of sample points are generated in search space using the LHM. Then, Calculate the fitness values of all points, based on the true fitness function and hence update the fitness function evaluations. Surrogate models are (statistical) model is then built to approximate the actual model. Once the RBF surrogate model is built, this cheap model solution is evaluated. Once the most promising solution is found, it is evaluated using the expensive fitness function and the ensemble is updated. This step followed by the second search process using the DE, which aims to find the global solution of the system under consideration. Figure 26 depicts the flowchart of the RBF-DE proposed algorithm. The steps of the proposed algorithm used in this study are as below.

1) Generate a set of design data points, $x_i, \quad i = 1, \dots, n$ over the design space;

$$x_i = \{x_1, x_{2,} \dots, x_n\} \ , \quad x_i \in s^p \tag{4.5}$$

Where *p* represents the number of design variables, *n* is the number of initial design data points, and *S* stands for the entire design space (*n* will be a function of *p*, increasing with the problem dimension).

2) Evaluate the values of the objective function and constraints using the selected design points;

$$Y_{\min} = \min \{f(x): (x_i, f(x_i)) \in S\} \tag{4.6}$$

Where $Y_{\min}$ represents the minimum of expensive function values.

3) Construct the RBF model over the design space using the expensive points evaluated in the step 2.
4) Evaluate the RBF SM and find the points with the smallest value among the points of RBF SM and save them in the database.
5) Run the DE algorithm and use the points from the database as initial population size to find the global solution
6) If a present stopping criterion is met, output the best solution in the database; otherwise go to step 2.

The basic idea of using surrogate models in the RBF-DE optimization process can be quite simple. First of all, several sample points will be obtained by the LHS to sample in the design space; second, the RBF SM for the object function is built with sufficient accuracy; third, the optimum is found by the DE optimizer, with the object function evaluated by RBF SM, rather than by the expensive analysis code. Since prediction with the RBF SM is much more efficient than that of the expensive analysis code, the optimization efficiency can be greatly improved. The compared results of the DE conventional optimization and DE surrogate-based optimization are drawn in Table 6. In addition to improved optimization efficiency, surrogate models also serve as an interface between the analysis code and the optimizer, which makes the establishment of an optimization procedure much easier. We can see from the proposed method that only a few evaluations of the expensive function of each iteration will be evaluated using the real objective function, except in the initialization process, in which all individuals in the population are fitness evaluated.

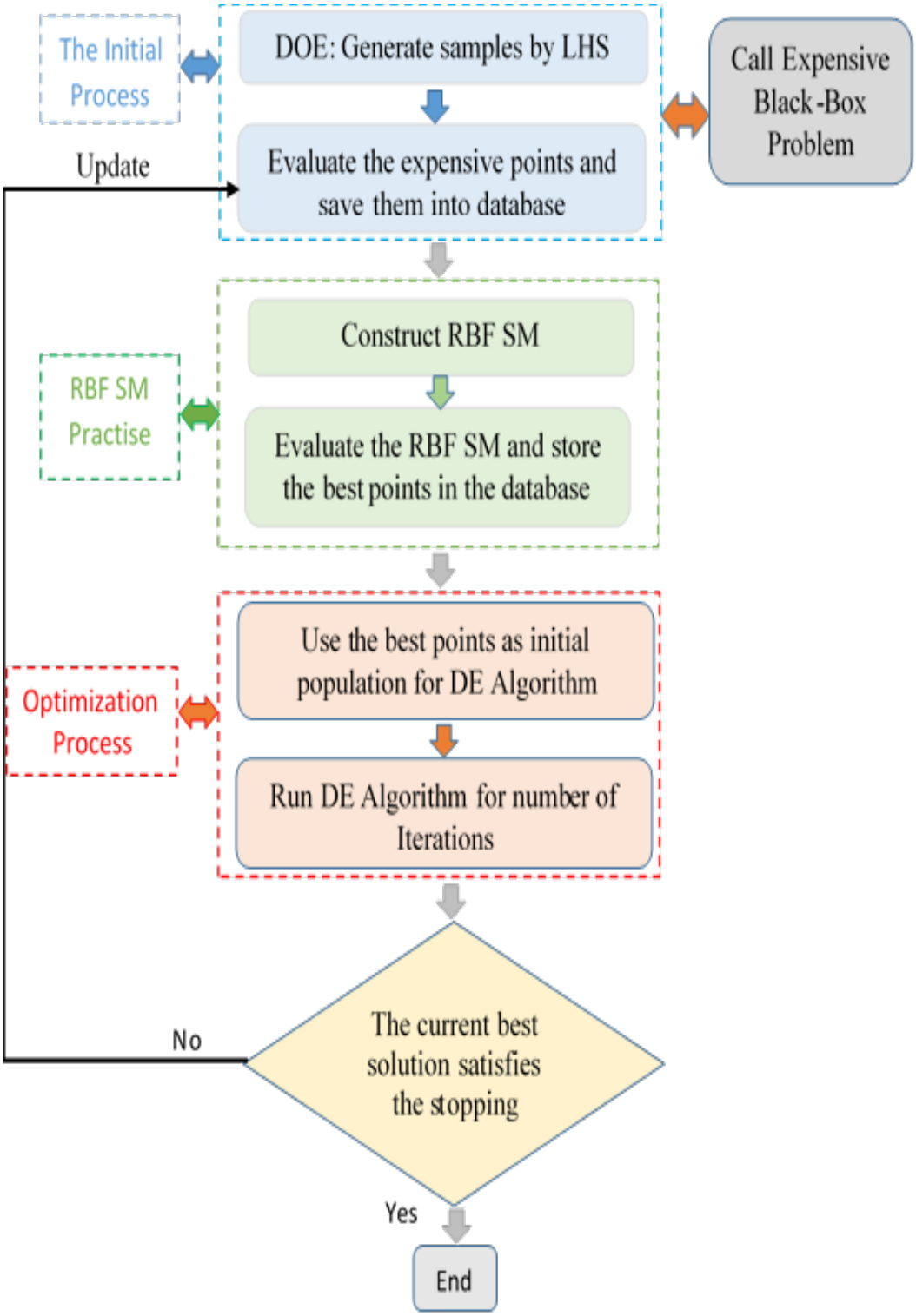


Figure 26. RBF-DE proposed method Flowchart

4.6 Numerical experiments using Benchmark functions

In order to evaluate the performance and show the efficiency and superiority of the proposed algorithm (RBF-DE), twelve well-known benchmark test functions mentioned in [104] are used ranging from 2 to 16 design variables. All these functions are minimization problems. Each benchmark function has its own properties – unimodal, multimodal, separable or non-separable as presented in Figure 27. It is noteworthy that the combination of these properties determines the complexity of the functions. A function is considered multimodal if it has two or more local optima, and it is considered separable if it can be rewritten as a sum of functions from just one variable. Table 5 presents the list of benchmark functions utilized to assess the performance of the evolutionary methods considered. The table consists of the name of the benchmark function, the range, the dimension, the characteristic of the function, and its formula. The characteristics of the function determine the complexity of the function shown in Appendix A. These Benchmark test functions have been used to evaluate the robustness, capability, and efficiency of the proposed RBF-DE algorithm.

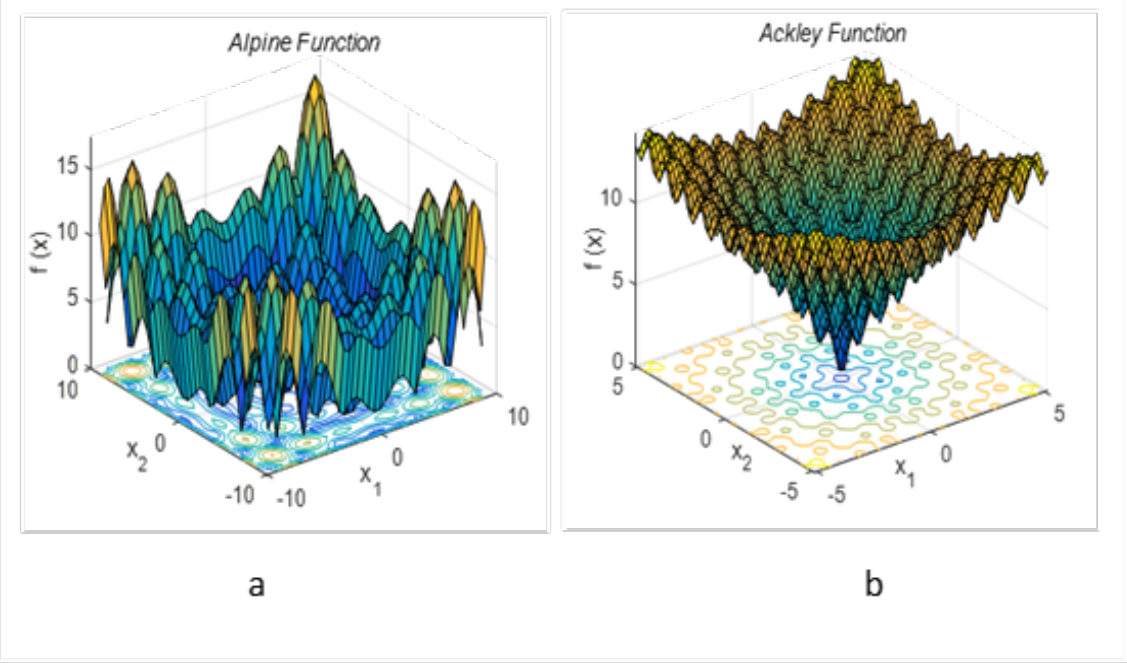


Figure 27. Samples of Unimodal and Multimodal Functions

Table 5: Benchmark test functions

No.	Function	D	Region	Analytic f*	Fun. Properties
f1	*Banana*	*2*	*[-2 , 2]*	*0.000*	*U-Modal*
f2	*SC*	*2*	*[-2 , 2]*	*-1.0316*	*M-Modal*
f3	*Beale*	*2*	*[-4.5 , 4.5]*	*0.000*	*M-Modal*
f4	*Peaks*	*2*	*[-3 , 4]*	*-6.5511*	*M-Modal*
f5	*Shubert*	*2*	*[-10 , 10]*	*-186.7309*	*M-Modal*
f6	*Shekel*	*4*	*[0 , 10]*	*-10.1532*	*M-Modal*
f7	*Ackley*	*5*	*[-35 , 35]*	*0.000*	*M-Modal*
f8	*Levy*	*5*	*[-10 , 10]*	*0.000*	*M-Modal*
f9	*HM 6*	*6*	*[-1 , 1]*	*-3.3220*	*M-Modal*
10	*Sum Squares*	*8*	*[-10 , 10]*	*0.000*	*U-Modal*
f11	*Sphere*	*10*	*[-5.12 , 5.12]*	*0.000*	*U-Modal*
f12	*HM 16*	*16*	*[-1 , 1]*	*25.8750*	*M-Modal*

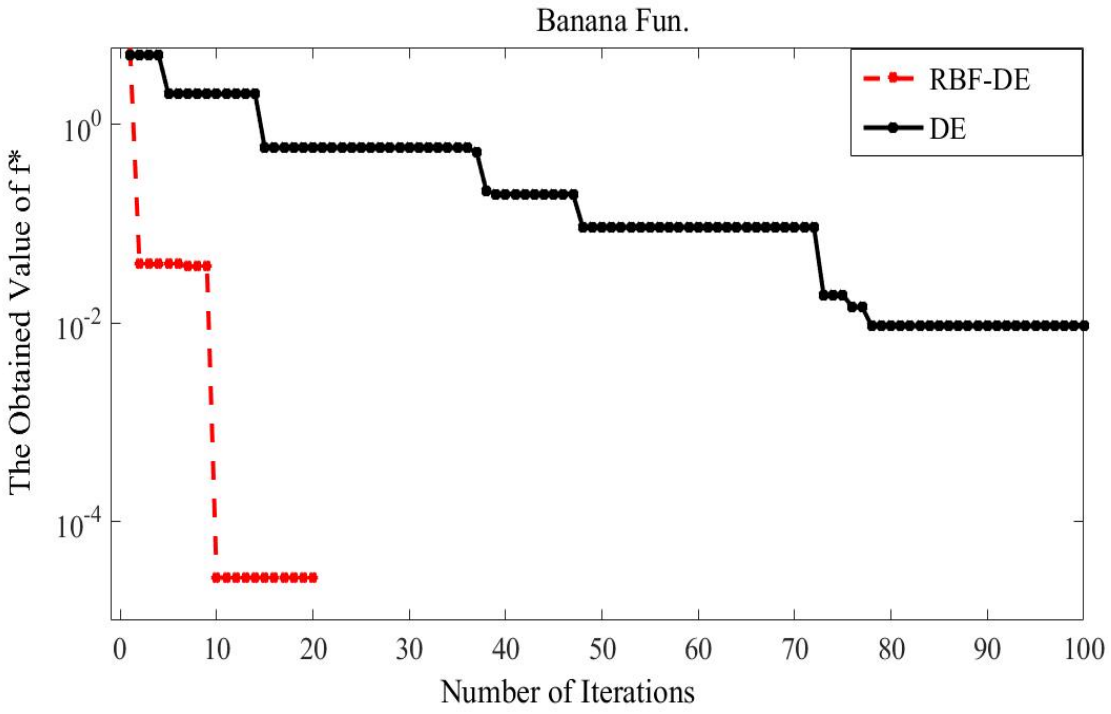


Figure 28. Test Function # 1

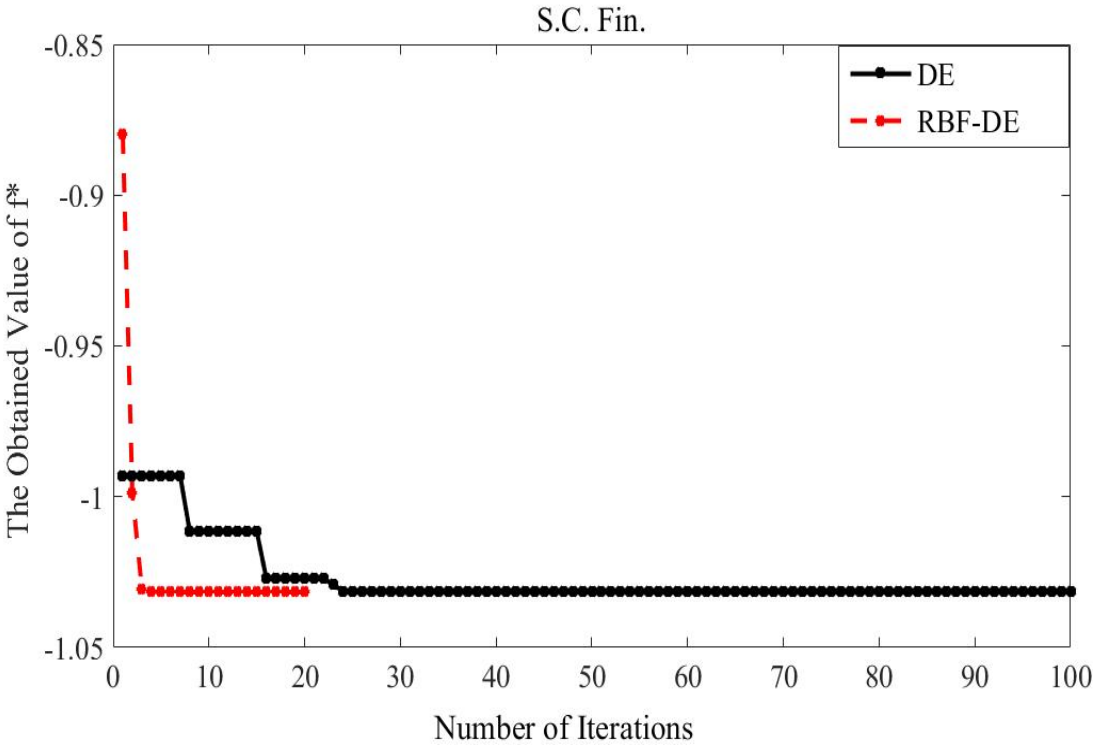


Figure 29. Test Function # 2

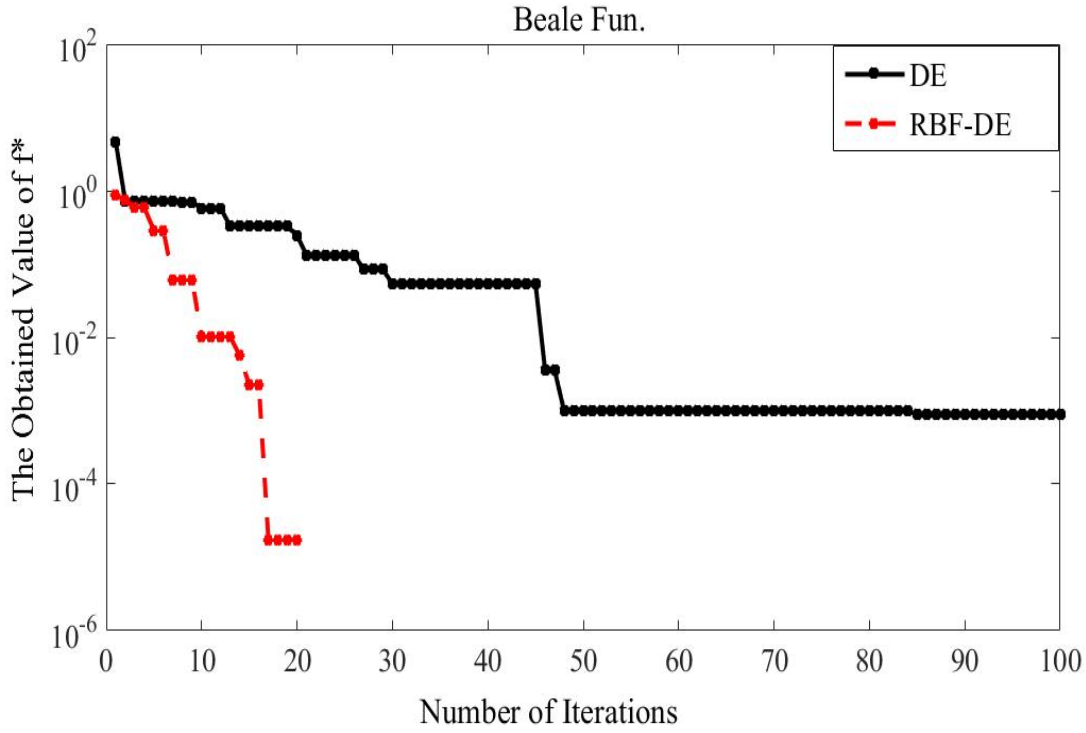


Figure 30. Test Function # 3

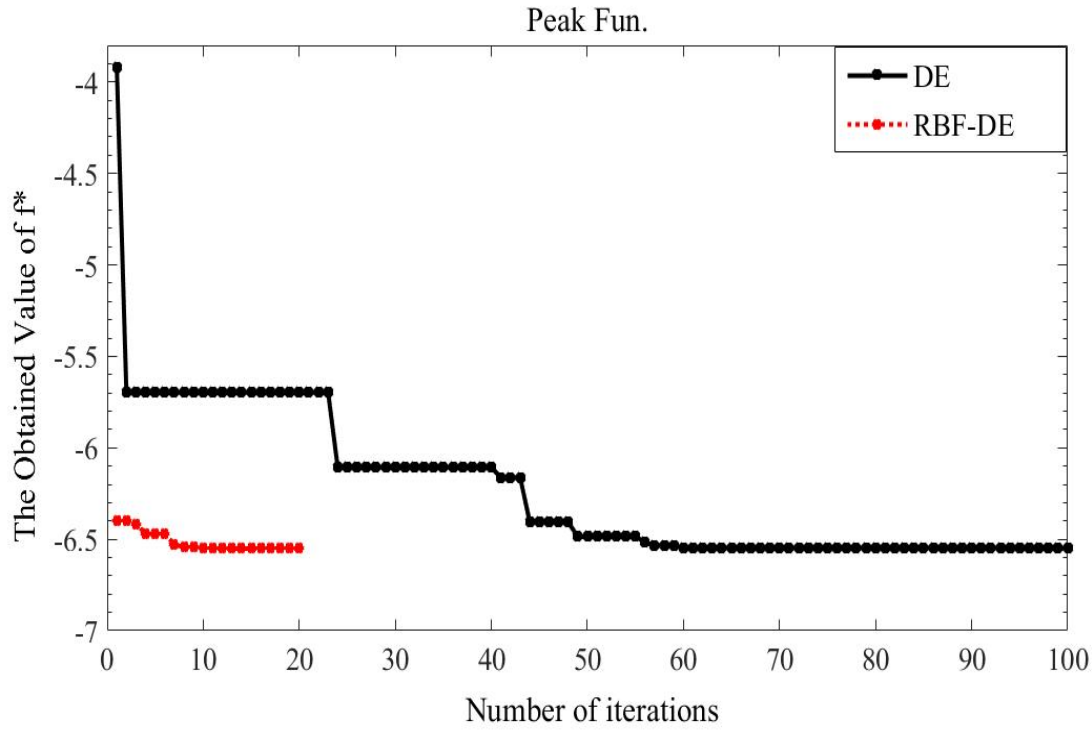


Figure 31. Test Function # 4

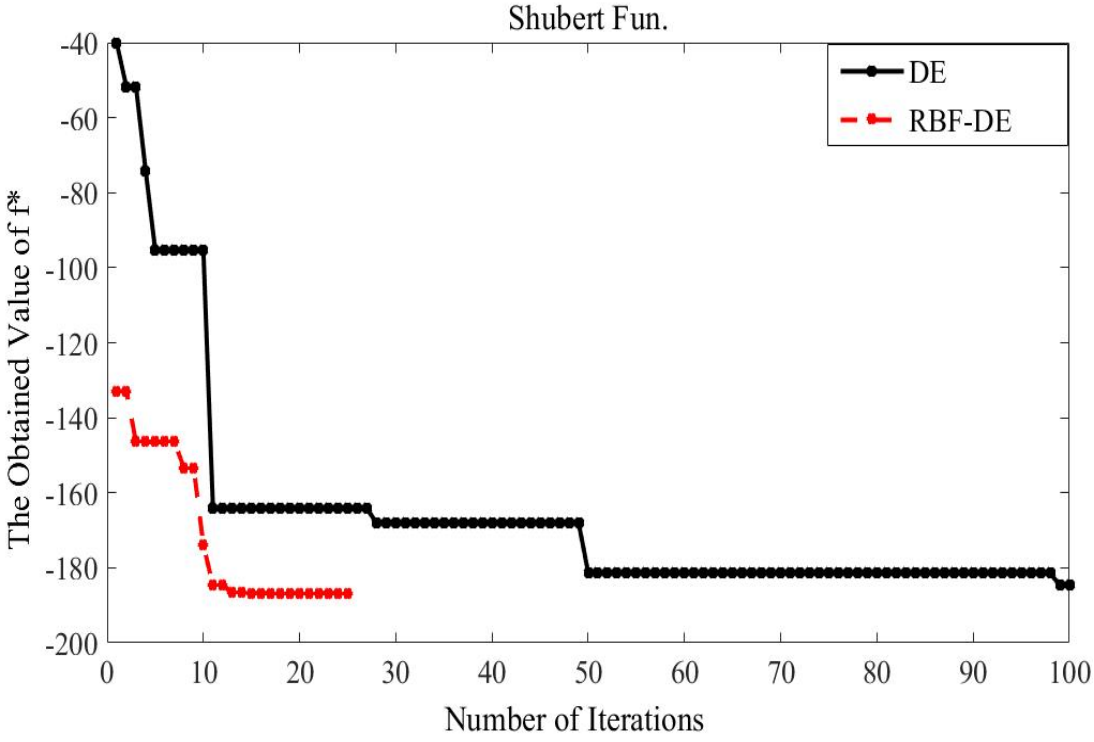


Figure 32. Test Function # 5

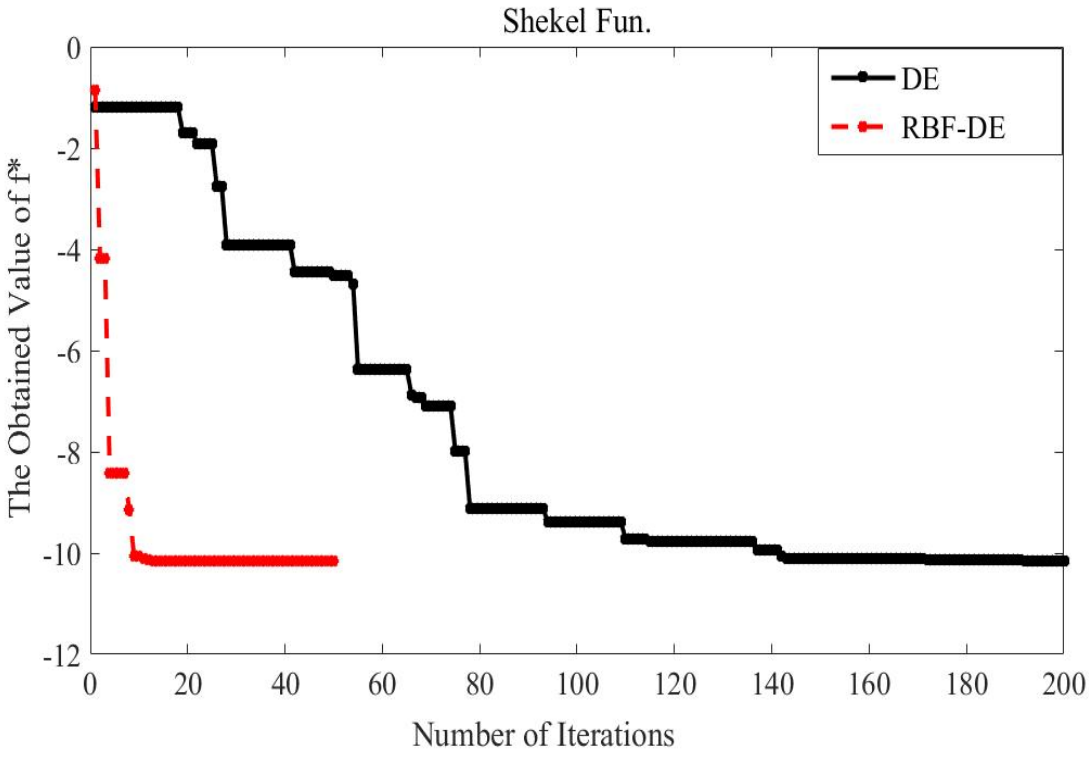


Figure 33. Test Function # 6

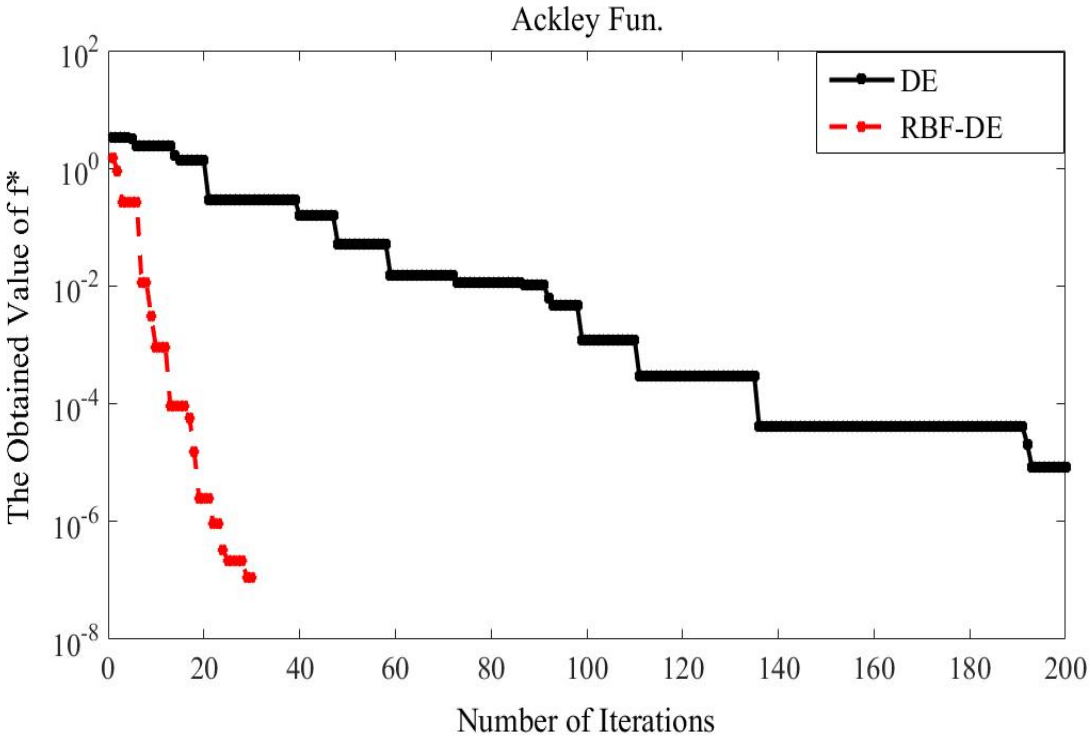


Figure 34. Test Function # 7

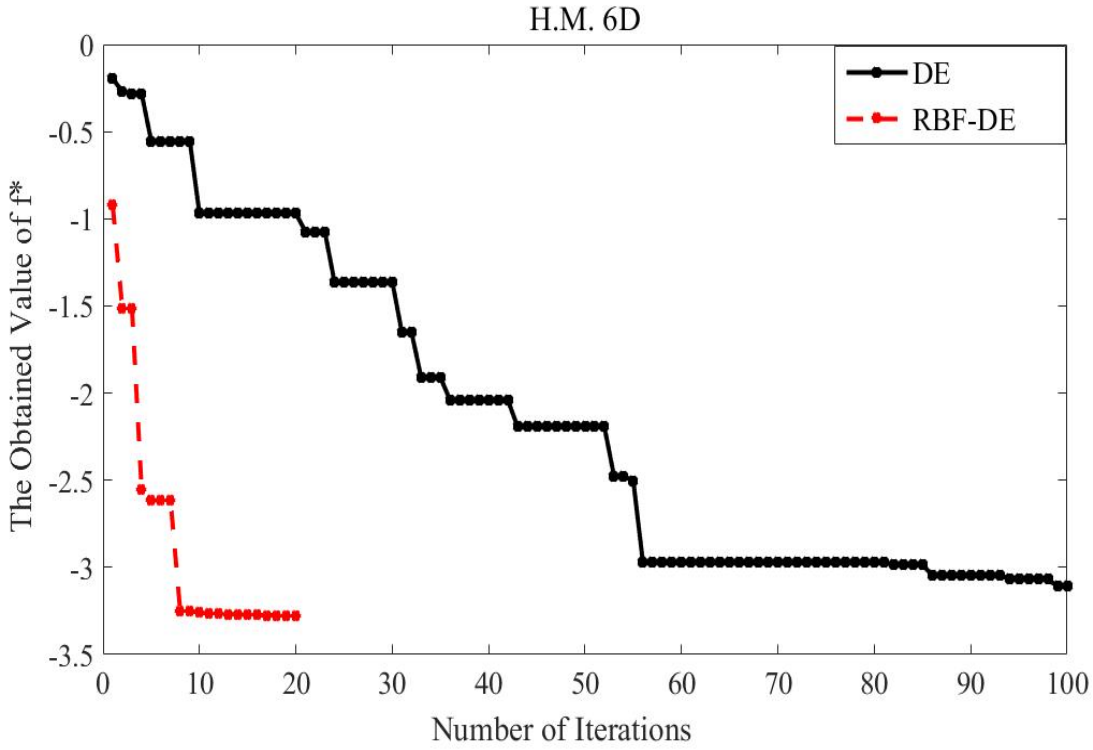


Figure 35. Test Function # 8

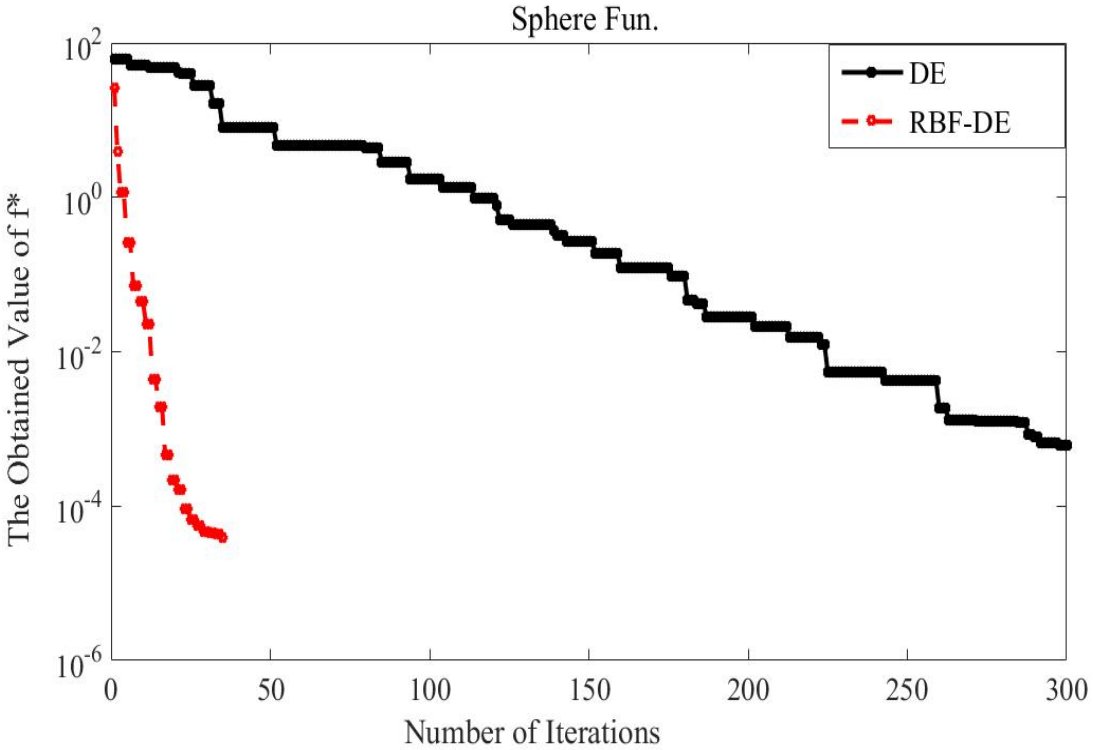


Figure 36. Test Function # 11

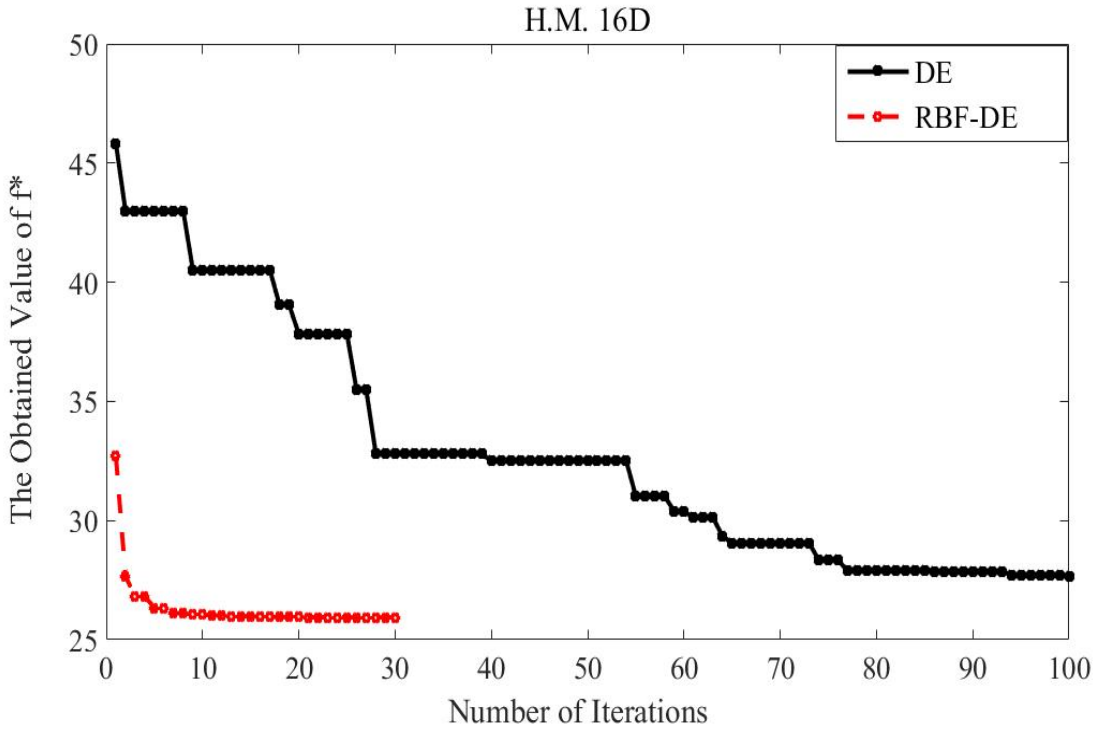


Figure 37. Test Function # 12

4.7 Experimental Results and Discussions

The principal objective of this work is to increase the performance of the DE algorithm for expensive black-box problems. A series of modifications have been proposed and RBF-DE can find the optimum solution with remarkably fewer NFE. In order to prove this assertion, the performance of the DE algorithm has been compared to the modified version using twelve standard benchmark functions with different dimensional test problems. Table 6 shows a summary of these results. It is notable that for a fair comparison, the same stopping criteria have been used for the original DE algorithm. The history of convergence in both methods can clarify the effects of the proposed modification. Results from Figure 28 to Figure 37 show the convergence rate of 2, 4, 5, 6, 10, and 16 variables. One can see that, in order to reach the same accuracy, the NFE required by the RBF-DE is significantly smaller than that for the original DE algorithm. The first benchmark function in Table 5 and Table 6 (e.g., Banana) is unimodal with a theoretical minimization value of zero. In f_1 (Banana), the result which is closest to the theoretical optimal value acquired by the RBF-DE is 2.372E-05 with only 242 NFE, and the DE can reach the best value of 6.999E-05 and 2500 NFE. In f_2 (SC) which is a multimodal function, the RBF-DE proposed algorithm

achieved the best minimization performance with a value of -1.0316 with 234 NFE, and DE has the second best value of -1.0315 with 1750 NFE. The next ten functions in these tables are mixed between unimodal and multimodal (Beale, Peak, Shubert, Shekel, Ackley, Levy, HM6, Sum Squares, Sphere and HM 16). The results in Beale function indicated that the RBF-DE method has achieved the optimal value of 1.731E-05 with only 1382 NFE. On the other hand, DE reached 3.371E-05 with 1750 NFE. RBF-DE is the best performing method on the f_7 (Ackley function) with the mean value of 2.36E-07 with 615 NFE followed by DE with the mean value of 3.89E-05 with 8500 NFE to reach the approximation global solution. In sphere function, both RBF-DE and DE provide good performance with 3.955E-05 and 3.682E-05 mean values. However, RBF-DE algorithm provides the same accuracy as DE with only 18 % of NFE required by the DE algorithm. Considering the results presented in Table 6, the RBF-DE appears to be the best overall performing approach, outperforming the original DE method in all twelve benchmark tested functions in terms of the NFE required to achieve almost the same accuracy. However, in terms of time consumed to complete the benchmark test, the DE is the best with an average for all twelve benchmark functions of 0.8850 seconds. In each case, the horizontal axis shows the functions as illustrated in Figure 38, while the vertical axis demonstrates its corresponding NFE. It is evident from these graphs that the RBF-DE proposed method not only decreases the required NFE, but also gives the user the opportunity to reach more accurate solutions at much lower number of samples. The main focus of this modification was to increase the performance of the DE algorithm on expensive black-box problems. Notice that the major purpose is not intended to beat the DE method proposed earlier, but to show that the RBF-DE algorithm can effectively serve as an attractive method to successfully handle CEBB problems.

Table 6: Comparison results of DE vs. RBF-DE

Fun	DE			RBF-DE			N.F.E Reduction
	Obt. *f**	N.F.E	CPU time	Obt. *f**	N.F.E	CPU time	
f_1	*6.999E-05*	*2500*	*1.9844*	*2.372E-05*	*242*	*2.0469*	*90.3 %*
f_2	*-1.0315*	*1750*	*1.6094*	*-1.0316*	*234*	*1.6875*	*86.6 %*
f_3	*3.371E-05*	*2500*	*0.5625*	*1.731E-05*	*382*	*4.25*	*84.7 %*
f_4	*-6.5511*	*2000*	*0.0625*	*-6.5511*	*325*	*3.6094*	*83.7 %*
f_5	*-186.0451*	*2500*	*0.375*	*-186.7309*	*272*	*6.7813*	*89.1 %*
f_6	*-10.1509*	*5000*	*0.25*	*-10.1532*	*876*	*20.734*	*82.5 %*
f_7	*3.89E-05*	*8500*	*0.40625*	*2.36E-07*	*615*	*11.0938*	*95.1 %*
f_8	*1.894E-05*	*7500*	*0.28125*	*3.808E-05*	*881*	*22.7344*	*88.2 %*
f_9	*-3.2201*	*2500*	*0.09375*	*-3.3259*	*239*	*2.625*	*90.4 %*
f_{10}	*1.593E-04*	*3750*	*0.1875*	*1.648E-04*	238	*1.7344*	*93.6 %*
f_{11}	*3.682E-05*	*8750*	*0.5625*	*3.955E-05*	*568*	*14.984*	*93.5 %*
f_{12}	*25.9383*	*5000*	*0.3125*	*25.9297*	*537*	*12.4844*	*89.3 %*

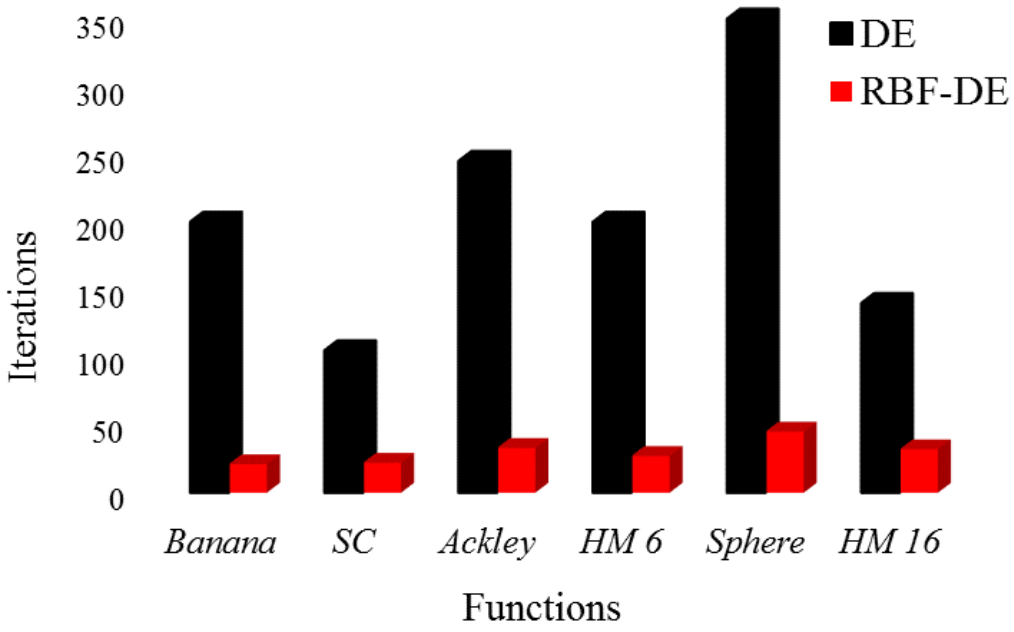


Figure 38. NFE used by DE Vs RBF-DE for number of benchmark functions

4.8 Summary

This Chapter presents the contributions that have been made in metamodeling for computationally expensive optimization problems. As mentioned in Chapter Two, the RBF is identified as a mature metamodeling technique which works in computationally expensive optimization problems. The DE algorithm is found to be costly for expensive black-box problems with an exponentially increasing demand for function evaluations. In this work, modification of the original DE was achieved by integrating the RBF model with DE; progressive reduction on the search region is evident. The proposed RBF-DE has been benchmarked using four standard tests, and the performance increase has been illustrated and discussed. At this point, it is to be noted that the exponentially increasing demand of DE algorithm for function evaluations in expensive black-box problems has been replaced with the new strategy. This makes the RBF-DE method a suitable choice for high cost functions although further improvements are needed to make it more efficient for computationally expansive black-box (CEBB) problems.

Chapter 5. A Comparative Study on Recently-Introduced Nature-Based Global Optimization Methods in Complex Mechanical System

5.1 Introduction

Advanced optimization methods are used in engineering design to obtain the best functional performance and/or minimum production cost of a complex product or system in the increasingly competitive international market. Nature-inspired global optimization algorithms with superior search efficiency and robustness have been continuously introduced and improved to solve various complex, nonlinear optimization problems, which most traditional gradient-based optimization methods are incapable to deal with. Moreover, these nature-inspired global optimization (GO) algorithms become more useful when the objective function of the problem of interest is in implicit, black-box form and/or its derivative information of is unavailable, unreliable, or expensive to obtain. Development of the nature-inspired global optimization algorithm dated back to the introductions of the GA based on Darwin's principle of biological systems by Holland *et al.* Since then, ACO, SA, and PSO, as the most recognized nature-inspired global optimization algorithms have been introduced. Due to their general applicability, ease of implantation, and relatively fast convergence, GA, PSO, ABC and SA have attracted much attention in the past, and have been applied successfully for a wide range of global optimization applications. For instance, Brenna *et al.* [105] employed GA optimizer to carry out multi-objective optimization for train schedules with minimum energy consumption and travel time; Zhao *et al.*[106] proposed an improved ant colony optimization (ACO) method for the route planning of the omnidirectional mobile vehicle; and Herish *et al.*[107] applied PSO in multi-objective optimization for reliability-redundancy assignments in design.

Over the years, significant efforts have been devoted to the further developments of various global optimization methods of both deterministic and stochastic types. The deterministic approach solves an optimization problem through a predetermined sequence of searches and search points, converging to the same or very close global optimum. Direct search [108], Branch and Bound [109], Clustering [110], and Tunnelling methods [111] are typical

examples of this approach. Stochastic methods, including nature-inspired techniques, are based on randomly sampled search points. Therefore, their different runs may result in different optimization results due to the random nature of the search steps. These stochastic search algorithms have the capability to identify the global optimum efficiently for many optimization problems, simply using the evaluated values of the objective function without the need of its gradient information.

With the advance of computation hardware and software, computational intensive, numerical analysis and simulation tools, such as Finite Element Analysis (FEA) and Computational Fluid Dynamics (CFD), became commonly used means for evaluating the performance of a new design with many design parameters. Global optimization can be used to search through the design space of these implicit and multimodal objective functions to identify the optimal design solution in principle. However, these types of computationally expensive, black-box function optimization problems require extensive computation during each evaluation of its objective function. Conventional nature-inspired optimization algorithms, such as GA and SA, which require very large number of objective function evaluations with lengthy computation time, particularly for high-dimensional problems with many design variables, become impractical and infeasible to use. To address this issue, a number of recently introduced nature-inspired optimization methods, such as the Artificial Bee Colony (ABC)[112], Firefly Algorithm (FFA) [113], Cuckoo Search (CS) [114], Bat Algorithm (BA) [115], Flower Pollination Algorithm (FPA) [116], and Grey Wolf Optimizer (GWO) [117], have been introduced with improved ability to deal with complex and high-dimensional global optimization problems [118].

This study tests and compares the performance of the aforementioned six nature-inspired GO search methods, BA, CSA, FFA, FPA, ABC and GWO, in terms of their capability of locating the true global optimum, search efficiency, and robustness in dealing with high-dimensional and computationally expensive black-box GO problems. These advanced GO methods are closely examined using a variety of test problems, ranging from benchmark test functions to real-life design optimization of complex mechanical systems. This Chapter begins with an overview on these relatively new algorithms, followed by a discussion on the evaluation criteria. The convergence process and results are discussed comprehensively to rank their search efficiency and robustness. The purpose of this work is firstly to identify

the most suitable GO algorithm to improve with the addition of SM, and secondly to introduce a real engineering design optimization problem as an example of computationally-expensive black-box global optimization. The test case can thus be used in the work presented in the following Chapter.

5.2 Nature-Inspired Global Optimization Methods

Nature-inspired Global Optimization approaches are a class of population-based methods commonly used for solving many practical global optimization problems including: GA, PSO, SA and ACO. However, these traditional GO algorithms need an enormous number of function evaluations during the search process in order to recognize the solution, and the number of search evaluations dramatically increases when the optimization problem becomes a high-dimensional problem. Hence, new effective and capable GO search methods become necessary. In recent years, a number of new search methods have been presented including the Artificial Bee Colony (ABC), the Firefly Algorithm (FFA), the Cuckoo Search (CS), the Bat Algorithm (BA), the Flower Pollination Algorithm (FPA), and the Grey Wolf Optimizer (GWO). In this section, the selected nature-inspired global optimization methods are discussed and explored in more detail. The pros and cons, as well as the applications of the six methods are also highlighted in this section. The methods have been chosen based on:

- They have been introduced in the last decade.
- They are widely used in solving many engineering globe optimization problems.
- They share many similarities in general. For instance, all these methods begin with a randomly population group and operate a fitness value to evaluate this population. They all update the population and randomly search for the optimum. They use a sharing information mechanism wherein the evolution only looks for the best solution.
- They have the potential to solve high-dimensional complex design problems especially when the number of iterations are limited.

5.2.1 Artificial Bee Colony Method

The Artificial Bee Colony (ABC) method, presented by Dervis Karaboga in 2005 [118], was inspired by the search behaviour of bee colonies. Honeybees use several mechanisms,

such as a waggle dance, to locate food sources. The waggle dance technique is used by scout bees to share information about the source of food. Bees can quickly and precisely modify their search pattern and search space in accordance to the updated information received from other bees. In the ABC method, artificial bees are classified into three different groups: employed bees, onlooker bees, and lastly, scout bees. Each of these groups is assigned different tasks in the ABC procedure. The employed bee's task is to visit and collect information about a source of nectar and then share it with other bees in the hive. In the ABC algorithm, the number of employed bees and the number of food sources are equal since each employed bee is linked with only one source of nectar. The onlooker bees receive the data regarding the sources of nectar from the employed bees, and then the onlooker bees select one of the sources and start to transfer the food to the hive. The scout bees fly in random directions and have the responsibility of finding new sources of nectar. In the ABC method, the possible solution corresponds to the nectar source location in which the fitness of the solution mainly depends on the related nectar amount. Furthermore, the total number of best obtained solutions is also identical to the number of employed bees or equivalent onlooker bees.

In the ABC method, a number of parameters including population size $\boldsymbol{N}$, number of cycles (iterations), and the exploration parameter limit should be set prior to running the optimization. The number of cycles (iterations) and the parameter limits are equally important, and, to obtain the best results, should be set to their optimal values. It must be noted that in the ABC method, employed bees apply a local search to each nectar source, whereas the onlooker bees will likely update better food sources. Therefore, in the ABC algorithm, the employed bees are responsible for diversification whereas the onlooker bees are used for condensation. By implementing these three groups of bees (employed, onlooker and scout), the ABC algorithm easily escapes from minimums and improves its search efficiency. The ABC algorithm is competitive with other nature-inspired algorithms [119]; its implementation is relatively easy, and it requires only a few tuning control parameters to be set. Due to the high efficiency of the ABC method, many researchers have used their own points of view to utilize ABC for different purposes. For instance, Gao and Liu [120], inspired by differential evolution (DE), developed a search strategy ABC/best/1 with a new chaotic initialization method with a subsequent improvement of the exploitation

ability of ABC. Zhu and Kwong [121] developed a global best- (gbest-) guided ABC algorithm (GABC) to improve the search ability as well as the efficiency by using the gbest solution with the original search formula of the ABC algorithm. Li *et al.* [122] presented a modified ABC method (I-ABC) using the best-so-far solution, inertia weight, and coefficients to improve the search efficiency. Banharnsakun *et al.* [123] presented an improved search equation that causes the solution to directly converge towards the best-so-far solution rather than through a randomly selected path. Xiang and An [124] developed an efficient and robust ABC algorithm (ERABC), in which a combined solution search equation is used to accelerate the search process. Xianneng and Yang [125] introduced a new ABC method with memory (ABCM) to guide the further foraging of the artificial bees by combining other search equations to select the best solution. Garg [126] used ABC to handle reliability-redundancy allocation optimization problem. The core steps of the ABC approach can be found in the given reference [127].

5.2.2 Firefly Algorithm Method

Xin-She Yang proposed the Firefly Algorithm (FFA) that was inspired by the flashing lights of fireflies. Almost all of the firefly species use flashing lights as an attraction communication signal between females and males. The recurring flash, the flashing rate, and the time delay between flashes are the key concepts used among fireflies to share information. Although the mechanism of flashing lights in fireflies is still largely unclear, it is obvious that this signalling system helps fireflies find food, protect themselves, and attract their prey. FFA has become one of the most efficient global optimization algorithms and has been used in many real-life problems [128]. In FFA, the objective function is connected to the flashing light. In other words, the brightness is proportional to the best solution of an optimization problem. Brightness will assist fireflies to travel to shinier and more attractive positions in order to find the global best solution. There are three important rules in the FFA procedure:

- Fireflies are socially oriented insects, and regardless of their sex, all of them move towards more attractive and brighter fireflies.
- The amount of attraction of a firefly is proportionate to its brightness; hence, a firefly that has less brightness will travel toward one that has higher brightness. The

attractiveness decreases as the space from the other firefly rises since air absorbs light. If there is no brighter firefly in the vicinity, fireflies will travel randomly in the design space.

- The brightness (light intensity) of a firefly is specified by the objective function value of the optimization problem.

In the firefly method, there are two essential keys: the variation of the light density and the formulation of the attractiveness. FFA assumes that the attraction of a firefly is determined by its shine, which is always connected with the value of the objective function: $I(x) \propto f(x)$, where $I(x) is$ the brightness of a firefly at position x and $f(x)$ is the value of the objective function. β is defined as the attractiveness of a firefly i to firefly j, which is affected by the distance between them. β_0 is the attractiveness at $r = 0$, where r is the space between two fireflies. Coefficient γ is the fixed light absorption value. Over the iterations, fireflies converge to the local optimal solutions. The global solution can then be obtained by comparing the local solutions. Implementation of FFA is often easier than other nature-based GO algorithms such as PSO, BA and ACO. FFA is reasonably efficient in solving many continuous complex optimization problems that are challenging to other powerful GO algorithms such as GA and PSO [129].

The Firefly algorithm has gained much attention, and, due to its simplicity, many modifications on the FFA have been achieved. Some instances of the attention that the Firefly algorithm has received are: Bhushan and Pillai [130] compared the performance and the effectiveness of FFA in dealing with nonlinear optimization problems against GA. Farahani *et al.* [131] employed FFA in solving continuous practical global optimization problems in dynamic environments. Younes *et al.* [132] implemented a hybrid FFA for solving multi-objective continuous/discrete GO problems. Talatahari *et al.* [133] used FFA to find the optimum design of structure design problems. Hassanzadeh *et al.* [134] modified FFA to deal with image processing optimization problems. Jati [135] employed FFA to solve many complicated GO problems such as TSP. Arora and Singh [136] considered the optimal choice range of FFA parameters in different numeric experiments. Bidar and Rashidy [137] improved FFA by adjusting the parameter controller in order to balance the exploitation and exploration of the algorithm. Gandomi *et al.* [138] introduced a hybrid algorithm by combining the Fuzzy C-Means (FCM) with FFA to improve the

clustering accuracy with global optimum solutions. Farahani *et al.* [139] increased the efficiency of FFA by stabilizing the algorithm's movement to direct the fireflies towards the global best if there is no better solution found.

5.2.3 Cuckoo Search Method

Another recent nature-inspired global optimization method is the Cuckoo Search (CS) approach developed by Yang in 2009 [140]. The CS method is based on the natural obligatory brood parasitic behaviour of cuckoo birds in integration with the Lévy flight. A cuckoo bird places its eggs in another bird's nest to be brooded by the mother bird of another species. In some cases, other birds engage in battle with the stranger cuckoos when the other bird realizes that the eggs in her nest are not her own. In this case, the other bird either destroys the unwelcome eggs in the nest or leaves its own nest and rebuilds a new one elsewhere. Some cuckoo female species have developed a new strategy based on imitating the colours and shapes of the eggs of other birds to increase the chance of reproduction and decrease the probability of desertion by the other bird. In general, the cuckoo's eggs hatch before the other bird's eggs, thus the first job of the cuckoo chick is to get rid of the other bird's eggs to increase its own chance of being fed by the resident mother bird. This knowledge of the Cuckoo bird has been used to develop the CS algorithm.

The easiest way of applying the CS algorithm is achieved through the following three steps [141]. First, every cuckoo bird places only one egg at a time in a random nest. Second, the best nests (solutions) with a good quality of eggs are selected for the next population. Third, the number of nests is constant, and the egg deposited by a cuckoo is recognized by the other bird with a probability of $P_a \in [0,1]$. Therefore, the other bird may either destroy the alien eggs or relinquish the nest and establish a new nest. The last assumption can be estimated as the fraction P_a of the n nests when new nests (completely new solutions) are substituted. In the CS method, every egg in a nest expresses a solution, and each cuckoo can only deposit one egg. This algorithm can also be used when the problem is more complex such as where each nest could hold several eggs representing a number of solutions. Further, each cuckoo can be simply considered as a random point in the design

space while the nest is the memories that are used to keep the previous solutions and compare them with the next solutions.

CS is used in dealing with high-dimensional, linear and nonlinear GO problems. A recent study showed that CS is more effective and robust than PSO and GA in multi-modal objective functions [141]. This is partially due to that there are only a limited number of parameters to adjust in the CS method compared to other GO algorithms such as PSO and GA. A comprehensive description of the structure of the CS method is available in [141]. Since the development of the CS method in 2010, several studies have been introduced to improve its performance. Walton *et al.* [142] modified the CS algorithm to be more effective in handling nonlinear GO problems such as mesh generation. Yildiz [143] employed the CS algorithm to find the optimal parameters for a machine in the milling process. Vazquez [144] used the CS algorithm with artificial neural network model can to deal with different linear and non-linear problems. Kaveh and Bakhshpoori [145] applied the CS algorithm in designing steel frames. Chifu *et al.* [146] used a modification of the CS algorithm to optimize the semantic web service. Tein and Ramli [147] proposed a discrete CS algorithm to solve nurse scheduling problems. Choudhary and Purohit [148] applied the CS algorithm to solve software data generation problems. Bulatovic' *et al.* [149] applied the CS algorithm in handling a six-bar optimization problem. Speed [150] modified the CS algorithm to be used efficiently in dealing with large-scale problems.

5.2.4 Bat Algorithm Method

The Bat Algorithm (BA) is a mature nature-based algorithm proposed by Yang based on prey tracking behaviour [151]. Using the concept of echolocation, bats create sounds while flying about hunting for food. These sounds are reflected to the bat providing useful information about the targets. This mechanism enables bats to identify the type of the objects, the distance from the target, and the kind and speed of the prey. Bats have the capability to establish three-dimensional pictures around the hunting area using their advanced echolocation strategy. While hunting, bats move randomly with velocity v_i at location x_i sending pulses with a range of frequency $f \in [f_{min}, f_{max}]$ (wavelength of λ and loudness A_0) while searching for prey. Bats can control the frequency pulses and regulate the rate of pulse emissions $r \in [0, 1]$ where 0 expresses that there are no emissions, and

1 expresses that the emissions of bats are at their maximum power. In BA, the loudness can be controlled from maximum (positive) A_0 to the lowest value. Note that $A_0 = 0$ expresses that a bat has reached its target and has stopped releasing any sounds. In BA, x^* is the best global solution at present and is obtained from among all achieved solutions. The value of f depends on the size of the design space. The higher the frequency, the shorter the wavelength and the shorter the travelled distance. Bats use a limited range of frequencies from 200 to 500 kHz. The steps of BA are described in [152].

BA is known as a very robust and efficient method in dealing with many engineering optimization problems [153]. Although many publications on this algorithm exist, BA still attracts a great deal of interest from researchers in a wide range of applications. Many researchers studied BA to ensure that it is able to avoid becoming trapped into local minima. For instance, Xie *et al.* [154] introduced a combination of Lévy flight with the BA (DLBA). Lin *et al.* [155] presented another hybrid of Lévy flight and bat approach (CLBA) for parameter approximation in a nonlinear dynamic model. Yılmaz and Kucuksille [156], motivated by the PSO algorithm and the ABC algorithm, proposed an improved bat algorithm (IBA) to advance the exploration mechanism of the algorithm. Wang and Guo [157] integrated the harmony search (HS) method into BA, and developed a hybrid metaheuristic (HSBA) method to increase the convergence speed of BA. Zhu *et al.* [158] improved the exploration capability of BA by modifying its equations. Afrabandpey *et al.* [152] introduced the chaotic sequences into BA in order to escape local convergence. Gandomi and Yang [159] replaced the four parameters in BA by different chaotic systems to increase the global search capability of BA. Kielkowicz and Grela [160] introduced some modification to the Bat Algorithm to solve nonlinear optimization.

5.2.5 Flower Pollination Algorithm Method

The Flower Pollination method (FPA) is stimulated by the nature of flower pollination. The main objective of flower pollination is reproduction. Pollination occurs in two main ways: abiotic and biotic. The majority of flowers use biotic pollination, where a pollinator such as a bee, a fly or a bird transfers pollen. In some cases, flowers and insects have special cooperation, where flowers catch the attention of and rely solely on a certain species of

insect to ensure an effective pollination process. Abiotic pollination is the result of wind and moving water and does not need any pollinators [161].

Self-pollination and cross-pollination are only two methods of flower pollination [162]. Self-pollination can happen when pollen is transferred from the same flower or to a different flower on the same plant when no pollinator is available. Therefore, this is considered to be a local solution. Cross-pollination happens when pollination transfers from the flowers of another plant. Biotic, cross-pollination can happen with long distance pollination and can be achieved by pollinators as a global solution. Those pollinators can behave as Lévy flight obeying Levy distribution steps. Hence, this behaviour was used to design FPA. In FPA, n is considered to be the flower pollen population, where k indicates the current iteration, and the parameter $L > 0$ is the density of the pollination. In FFA both local search and global search can be adjusted by changing the probability between $p \in [0, 1]$ in order to reach the optimum solution. The mechanisms described above are formulated into mathematical expression in order to solve any optimization problem of interest. FPA has many advantages in comparison to other nature-inspired algorithms - in particular its simplicity and flexibility. Moreover, FPA requires only a few tuning parameters making its implementation relatively easy. PA has been efficiency adapted to deal with a number of real-world design GO problems in engineering and science. Alam *et al.* [163] used FPA for solving problems in the area of solar PV parameter estimation. Dubey *et al.* [163] modified FPA to deal with the fuzzy selection of dynamic GO problems. Yang *et al.* [116] extended FPA to solve multi-objective GO problems, developing a multi-objective flower pollination algorithm (MOFPA). Henawy *et al.* [164] introduced a hybrid of FPA combined with harmony search (FPCHS) to increase the accuracy of FPA. Wang and Zhou [165] presented a new search strategy called a Dimension by Dimension Improvement of FPA (DDIFPA) to enhance the convergence speed and solution quality of the original FPA. Kanagasabai *et al.* [166] integrated FPA with PSO (FPAPSO) to deal with reactive power dispatch problems. Meng *et al.* [167] presented a modified flower pollination algorithm (MFPA) while attempting to optimize five mechanical engineering design problems. Binh *et al.* [168] proposed the Chaotic Flower Pollination Algorithm (CFPA) to overcome the large computation time and solution instability of FPA. Łukasik *et al.* [169] studied FPA in solving continuous global optimization problems in intelligent

systems. Sakib *et al.* [170] also presented a comparative study of FPA and BA in solving continuous global optimization problems. The main steps and the mathematical equations of FPA are explained in detail in [116].

5.2.6 Grey Wolf Optimizer Method

The Grey Wolf Optimizer (GWO) approach is a recently proposed algorithm and is based on the behaviour of grey wolves in the wild [117]. GWO simulates the leadership policy and hunting strategy of a grey wolf's family in its natural habitat. GWO is similar to other nature-inspired population-based approaches such as GA, PSO and ACO. In a family of grey wolves, there are four different groups: alpha, beta, delta and omega. The alpha, which is always male, is in charge of making decisions in hunting, selecting rest and sleeping places, etc., and its decisions must be obeyed by the rest of the family. Due to its dominating role, alpha is placed at the top of the family pyramid. Beta is at the second level in the family, and its duty is to support the alpha's decisions or other family initiatives. Beta can be female or male, and, because of its experience working alongside the alpha, can replace the alpha if it becomes necessary. Beta acts as a counsellor to the alpha and ensures that the alpha's orders are applied in the community, while at the same time; it guides the lower-level wolves. Further down the pyramid is the Delta that must follow the orders of the alpha and beta wolves but has domination over the omega. The duty of the delta is to defend and provide safety to all family members. Omega is the lowest ranked among the grey wolf family and plays the role of scapegoat. Omegas are the last group of the grey wolf family to eat from the prey, and its duty is to take care of the new-born pups. These three groups are used to simulate the leadership hierarchy in the grey wolf family. The first three groups lead omega wolves to search the space. During this search, all members update their positions according to the locations of the alpha, beta and delta.

In GWO, there are three steps of hunting which must be realized: finding prey, surrounding prey, and finally attacking and killing prey. Through the optimization procedure, the three most effective candidate solutions are alpha, beta and delta, as they are likely to be at the location of the optimal solution. Meanwhile the omega wolves must relocate with respect to the location of the other groups. As laid out in the GWO algorithm, alpha is the fittest candidate, but beta and delta gain better information about the potential position of prey

than omegas. Accordingly, the best three solutions are saved in the database, while the rest of the search agents (omegas) are obliged to update their position according to the position of the best solution so far. In GWO, n is the wolf population, k indicates the number of iteration, A and C are random parameters $A = (1,0), C = (1,1)$, x_p represents the location vector of the prey, x is the location of the agent, and a is a random value to update position. The GWO algorithm is recognized as being a capable and efficient optimization tool that can provide a very accurate result without becoming trapped in local optima [171]. Because of its inherent advantages, GWO is used in several optimal design applications. Kamboj *et al.* [172] successfully applied GWO in solving economic dispatch problems. Emary *et al.* [173] dealt with feature subset selection problems using GWO. Gholizadeh [174] utilized GWO to find the best design for nonlinear optimization problem of double layer grids. Yusof and Mustaffa [175] developed GWO to forecast daily crude oil prices. Komaki and Kayvanfar [176] employed GWO to solve scheduling optimization problems. El-Fergany and Hasanien [177] integrated GWO and DE to handle single and complex power flow problems. Zawbaa *et al.* [178] developed and applied a new version of GWO called the binary grey wolf optimization (BGWO) to find the optimal zone of the complex design space. Kohli and Arora [179] introduced the chaos theory into the GWO algorithm (CGWO) with the aim of accelerating its global convergence speed. Mittal *et al.* [180] proposed a modified grey wolf optimizer (MGWO) to improve the exploration and exploitation capability of the GWO that led to optimal efficiency of the method. GWO implementation steps are referenced in [117].

5.3 Benchmark Function and Experiment Materials

The most significant collections of global optimization benchmark problems can be found in [104, 117]. Each benchmark problem has its individual characteristic, such as whether it is a multimodal, a unimodal or a random shape function. The collection of the properties of these functions defines the complexity of benchmark problems. A problem is considered to be unimodal when it has only a local minimum which is also the global optimal, and it is classed as a multimodal when it has many local minima. The optimization problem is more difficult when the problem is non-convex or multimodal. In the search procedure, the neighbourhood around local minima should be avoided as much as possible so that the

optimizer is not stuck in local minima regions. If the problem has many local minima that are randomly distributed in the design space, the problem is extremely challenging. Another significant factor that defines the complexity of the optimization problem is the number of dimensions found in the design space. Table 7 lists benchmark problems employed for the efficiency evaluation of the above optimization algorithms. The table includes name, range, dimension, characteristics and the formulas of the benchmark functions are shown in Appendix A. The benchmark problems used in this work have been widely used in the literature to provide a deep understanding of the performance of these six nature-based optimization algorithms, see the Appendix A.

5.4 Experiments

Benchmark functions are used to assess the robustness and effectiveness of optimization algorithms. In this work, fifteen benchmark functions, as computationally expensive black-box functions with different properties and characteristics, are used to evaluate the efficiency of the aforementioned optimization approaches. In this regard, three experiments are performed. The first experiment compares the performance among the six discussed procedures (ABC, FFA, CS, BA, PFA, and GWO) while the iteration number is restricted. The second experiment tests the consequences of increasing the dimensions of the problem on the efficiency of the algorithm. The final experiment observes the sensitivity of the CPU time needed to obtain the global solution to the number of design variables (dimensional) change.

Table 7: Selected Benchmark Functions.

No.	Fun.	D	Search Space	Analytic f^*	Fun. Category
f_1	*Sphere*	*50*	$[-5.12, 5.12]^{50}$	*0.0000*	*U-Modal*
f_2	*Sargan*	*50*	$[-100, 100]^{50}$	*0.0000*	*U-Modal*
f_3	*S. Square*	*50*	$[-10, 10]^{50}$	*0.0000*	*U-Modal*
f_4	*Powell*	*50*	$[-4, 5]^{50}$	*0.0000*	*U-Modal*
f_5	*Schwefel*	*50*	$[-100, 100]^{50}$	*0.0000*	*U-Modal*
f_6	*Ackley*	*30*	$[-32, 32]^{30}$	*0.0000*	*M-Modal*
f_7	*Griewank*	*30*	$[-100, 100]^{30}$	*0.0000*	*M-Modal*
f_8	*Alpine*	*30*	$[-10, 10]^{30}$	*0.0000*	*M-Modal*
f_9	*Egg Carte*	*30*	$[-5, 5]^{30}$	*0.0000*	*M-Modal*
f_{10}	*Rastrigin*	*30*	$[-10, 10]^{30}$	*0.0000*	*M-Modal*
f_{11}	*Leon*	*25*	$[-1.2, 1.2]^{25}$	*0.0000*	*H. Converge*
f_{12}	*Zakharov*	*25*	$[-5, 10]^{25}$	*0.0000*	*H. Converge*
f_{13}	*Dixion &Price*	*25*	$[-5, 5]^{25}$	*0.0000*	*H. Converge*
f_{14}	*Cigar*	*25*	$[-100, 100]^{25}$	*0.0000*	*H. Converge*
f_{15}	*Levy*	*25*	$[-10, 10]^{25}$	*0.0000*	*H. Converge*

Experiment 1: The purpose of this experiment is to compare the optimization algorithms in terms of the efficiency when only a limited number of function evaluations (NFE) are available. In order to have unbiased comparisons, all approaches are restricted to the same iteration number (1000) as well as the same population size (40).

Experiment 2: The impact of increasing the number of variables on the resultant accuracy of the above algorithms is examined for fifteen high-dimensional benchmark functions considered as computationally expensive black-box problems. The selected benchmark problems with their properties are shown in Table 7.

Experiment 3: The computation time needed for solving computationally expensive black-box problems is the most important matter in assessing the efficiency of an algorithm. In this experiment, the above algorithms are also compared in terms of the computation (CPU) time required to reach the global solution on the same set of benchmark problems with different numbers of design variables (dimensions).

5.5 Setting Parameters in the Experiments

The setting of parameters is indeed an essential part of applying these algorithms, affecting the outcome of the search. It is imperative to set the parameters associated with each approach to its most appropriate values to obtain the best performance of the algorithms. Generally, these parameters are found prior to implementing the algorithm and remain unchanged during execution. Several studies indicate that the best mechanism for selecting parameters is based on intensive experiments to reach the optimal parameters of any algorithm [181]. Yang X-S [182] has conducted a number of runs to select the optimal setting parameters of BA. In the ABC algorithm, Akay and Karaboga [183] have suggested efficient parameters to obtain the best results of ABC. Yuan *et al.* [184] have recommended better parameter values in order to enhance the efficiency of FFA and converge the global optimum. Wang *et al.* [185] have provided the optimal setting parameters of CS to solve the problem of Chaotic Systems in order to gain the best performance of the CA method. Yang [186] has carried out parametric studies to obtain the best values for setting parameters of FPA. Rodríguez and Castillo [187] have proposed the optimal setting parameters of the GWO to achieve the best performance of the GWO approach. The setting parameters used in Table 8 of this work are selected based on the studies mentioned in this section.

Table 8: Setting Parameters Associated with Each Method

Algorithm	Setting Parameters
ABC	*Food Source = 20, the limit value = 10*
FFA	$\alpha = 0.5, \gamma = 1, \beta = 1$
CSA	$p = 0.25, \gamma = 1.5, \alpha = 0.01$
BA	$\alpha = 0.9$, *Population Loudness (A) = 0.25, Pulse rate* $F \epsilon$ *[0 2]*
FPA	$p = 0.8, \gamma = 0.1, \lambda = 1.5$
GWO	$\alpha \epsilon [0\ 2]$, $C \epsilon$ *[0 3]*

5.6 Experiments Results

In this section, the optimization methods are compared against each other in terms of efficiency, capability and computational complexity using the fifteen high-dimensional benchmark functions. Although the overall performance of an optimization algorithm changes depending on setting parameters and other experimental criteria, benchmark

problems can be used to indicate the efficiency of the algorithm under different levels of complexity. The statistical significance results from experiments on the selected benchmark functions are presented in Table 9 . From Figure 39 to Figure 47 only the results for the unconstrained benchmark functions with different surface with D = 25, 40 and 50. The outcome of this study may not reflect the overall efficiency of the tested algorithms under all conditions. All experiments are conducted using a PC with a processor running at 2.60 GHz and 16 GB RAM in MATLAB R2015b running under Windows 8.1. It should be mentioned that the results are based on 25 independent runs for functions $f1$ to $f15$, with a population size set at 40 and the maximum number of iterations fixed at 1000. In these experiments, the basic versions of the selected algorithms are considered without further modification. All methods codes are obtained from different sources, and adjusted to be harmonic with our work objectives.

5.7 Discussions

In this section, the statistical results obtained in this study are compared and discussed in detail broken down by the target criteria.

5.7.1 The Accuracy with Limited Number of Iterations

In this experiment, the impact of a limit of the number of iterations as well as the number of function evaluations (NFE) on the accuracy of the results of the algorithms are analyzed. Figure 39 to Figure 41 show the optimal solution found by BA, CSA, FFA, FPA, ABC and GWO methods versus the iteration number for three computationally expensive problems. The selected functions are Sphere (f_1), Griewank (f_7) and Cigar (f_{14}), which represent unimodal, multimodal and tough shape functions. The primary goal is to measure the performance of these algorithms based on different surface topologies/structures. The accuracy is measured by comparing the obtained optimum with the known actual optimum of the function. The evaluated efficiency is based on the convergence rate of these algorithms under the given conditions. The algorithm that yields accurate global solutions in a shorter computation time is the efficient and robust method.

As illustrated from Figure 39 to Figure 41 (i.e., Sphere, Griewank and Cigar), GWO, followed by ABC, have shown the best performance among the examined algorithms

regardless of the function form used. GWO starts to converge to the global solution within approximately 100 iterations, while the other algorithms require a higher number of iterations to find the region where the global solution may exist. The ABC algorithm also demonstrates its superior efficiency in terms of the accuracy of the solution. This is, perhaps, because of the GWO's and ABC's search mechanism that prevents them from easily being trapped in local optima. GWO reaches the global solution (6.14 E-56, 0 and 3.51E-101) respectively with a very smooth convergence rate, and ABC is the second best (5.60E-05, 1.96 E-10 and 2.14E-12) as shown in Table 9. On the other hand, BA and FPA show the lowest convergence rates with comparably low efficiencies. It appears that BA and FPA require a higher number of iterations to converge to the global solution.

Table 9: Summary of results for unconstrained optimization problems

Alg.	*f_1 (D = 50)*					*f_2 (D = 50)*					*f_3 (D = 50)*				
	Min	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**
BA	3.168	13.91	±8.94	20000	0.219	79160	829400	±2.74 X 10^{5}	20000	0.484	4242	5812	±210	20000	0.234
CSA	0.6846	5.443	±35.09	20000	0.375	14.06	2430	±1.72 X 10^{5}	20000	1.219	1.5841	29.54	±237	20000	0.719
FFA	1.1 X 10^{-3}	0.1129	±38.27	20000	1.266	0.1299	200.2	±2.16 X 10^{5}	20000	1.453	3.1861	13.67	±298	20000	1.266
FPA	7.373	10.10	±35.66	20000	0.344	187900	34570	±2.70 X 10^{5}	20000	0.547	1099	2518	±413	20000	0.313
ABC	5.60 X 10^{-5}	0.2723	±68.51	12170	0.172	4.5 X 10^{-3}	0.0321	±3.42 X 10^{2}	19956	0.449	5.60 X 10^{5}	16.18	±637	17534	0.172
CWO	6.14 X 10^{-56}	6.27 X 10^{-19}	±20.60	20000	0.297	0	4.76 X 10^{-19}	±1.49 X 10^{5}	20000	0.469	0	4.20 X 10^{-26}	±175	20000	0.281
	f5 (D = 50)					*f6 (D = 30)*									
	Min	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**
BA	11.92	147.1	±303.8	20000	0.50	1.44 X 10^{-7}	4.4 X 10^{-7}	±0.177	20000	0.234	16.77	16.77	±0.576	20000	0.25
CSA	6.38 X 10^{-2}	1.094	±1792	20000	1.109	5.03 X 10^{-43}	9.51 X 10^{-25}	±0.052	20000	0.406	4.567	6.99	±3.99	20000	0.375
FFA	2.644	5.696	±1332	20000	1.406	1.64 X 10^{-5}	1.77 X 10^{-4}	±0.078	20000	1.266	1.30 X 10^{-3}	9.61 X 10^{-2}	±1.51	20000	1.234
FFP	54.89	109.7	±2759	20000	0.578	0	2.17 X 10^{-13}	±0.128	20000	0.344	3.022	4.206	±4.268	20000	0.313
ABC	5.10 X 10^{-3}	0.631	±1356	20096	0.297	3.12 X 10^{-4}	3.45 X 10^{-2}	±0. 755	20005	0.156	5.79 X 10^{-5}	9.34 X 10^{-2}	±4.36	19645	0.188
GWO	1.23 X 10^{-6}	1.10 X 10^{-3}	±1190	20000	0.453	0	0	±0. 760	20000	0.303	7.99 X 10^{-15}	8.10 X 10^{-15}	±2.43	20000	0.203
	f8 (D = 30)					*f9 (D = 30)*									
	Min	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD.**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**
BA	6.867	6.960	±0.173	20000	0.266	2.53 X 10^{-7}	3.04 X 10^{-6}	±1.90 X 10^{-3}	20000	0.219	7.41X 10^{-14}	5.91X 10^{-12}	±1.10 X 10^{-6}	20000	1.391
CSA	0.1534	1.073	±2.647	20000	0.375	3.94 X 10^{-13}	2.74 X 10^{-10}	±2.80 X 10^{-4}	20000	0.375	1.34X 10^{-75}	1.70 X 10^{-34}	±0.166	20000	1.531
FFA	3.21 X 10^{-4}	0.2293	±1.922	20000	1.281	5.57 X 10^{-9}	1.31 X 10^{-6}	±1.90 X 10^{-4}	20000	1.5	9.81 X 10^{-14}	1.10 X 10^{-9}	±0.062	20000	2.188
FPA	1.1599	1.353	±1.455	20000	0.328	3.43 X 10^{-10}	3.08 X 10^{-8}	±1.17 X 10^{-2}	20000	0.344	1.06 X 10^{-36}	1.30X 10^{-21}	±1.80 X 10^{-6}	20000	1.469
ABC	1.96 X 10^{-10}	1.20 X 10^{-3}	±3.159	16890	0.203	5.50 X 10^{-6}	5.50 X 10^{-6}	±4.62 X 10^{-2}	18006	0.188	5.00 X 10^{-8}	1.90 X 10^{-5}	±0.153	20175	0.734
GWO	0	0	±0.996	20000	0.234	0	0	±0.9112	20000	0.219	0	0	±2.00 X 10^{-2}	20000	0.984
	f10 (D = 30)					*f11 (D = 25)*					*f12 (D = 25)*				
	Min	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**
BA	117.4	117.4	±34.98	20000	0.234	6.44 X 10^{-11}	6.45 X 10^{-11}	±4.50 X 10^{2}	20000	0.281	40.45	69.48	±22.80	20000	0.219
FFA	33.28	48.59	±44.13	20000	0.328	0	0	±8.80 X 10^{2}	20000	0.391	7.844	31.57	±5.41 X 10^{7}	20000	0.313
CSA	21.89	24.42	±53.68	20000	1.234	4.59 X 10^{-14}	2.43 X 10^{-9}	±7.10 X 10^{3}	20000	1.203	0.7202	3.704	±1.05 X 10^{5}	20000	1.125
FPA	62.22	94.78	±50.91	20000	0.297	2.42 X 10^{-28}	8.57 X 10^{-19}	±2.61 X 10^{2}	20000	0.359	15.71	40.18	±1.26 X 10^{5}	20000	0.281
ABC	1.93 X 10^{-9}	2.249	±3937	19700	0.161	3.71 X 10^{-5}	3.71 X 10^{-5}	±0.776	18765	0.219	105.9	133.4	±2.52 X 10^{7}	16742	0.166
GWO	0	0	±33.09	20000	0.291	1.10 X 10^{-6}	1.11 X 10^{-6}	±7.90 X 10^{2}	20000	0.217	3.85 X 10^{-38}	2.20 X 10^{-18}	±2.31 X 10^{7}	20000	0.154
	f13 (D = 25)					*f14 (D = 25)*					*f15 (D = 25)*				
	Min	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**	**Min**	**Median**	**SD**	**N.F.E**	**CPU**
BA	286.64	12600	±1.15 X 10^{4}	20000	0.219	49000	51900	±1.68 X 10^{4}	20000	0.344	8.881	23.88	±10.29	20000	0.221
CSA	4.9342	80.59	±3.84 X 10^{4}	20000	0.297	0.9062	460.3	±4.34 X 10^{5}	20000	0.469	2.05 X 10^{-5}	0.226	±15.44	20000	0.313
FFA	1.2766	1.095	±3.45 X 10^{4}	20000	1.094	3.91 X 10^{-2}	248.9	±3.79 X 10^{5}	20000	1.734	7.86 X 10^{-7}	3.30 X 10^{-3}	±11.21	20000	1.188
FPA	52.5473	314.1	±4.57 X 10^{4}	20000	0.281	7490	47800	±6.07 X 10^{5}	20000	0.406	2.977	6.616	±18.71	20000	0.313
ABC	9.84 X 10^{-4}	3.10 X 10^{-3}	±1.53 X 10^{3}	18900	0.188	2.14 X 10^{-12}	1.20 X 10^{-3}	±4.98 X 10^{5}	19150	0.25	1.43 X 10^{-15}	1.3 X 10^{-6}	±19.96	19406	0.256
GWO	0.6666	2.70 X 10^{-3}	±1. 10 X 10^{3}	20000	0.172	3.5110 $^{-101}$	5.05 X 10^{-45}	±2.41 X 10^{5}	20000	0.234	0.1418	0.1804	±6.38	20000	0.188

This may be because of the poor exploration ability of BA. In addition, the multidimensional objective problem can affect the convergence rate and the accuracy of the solutions obtained by PFA. This result implies that BA, FPA and CSA are potentially more efficient in solving low-dimensional computationally expensive optimization problems.

The performance of any algorithm typically depends upon how well the algorithm can balance the exploitation and exploration mechanisms. With intensive exploitation and poor exploration ability, algorithms such as BA easily are trapped into the local minima, whereas GWO and ABC show a much better performance, especially on the functions with a large number of local minima. CSA and FFA do not perform as well as GWO and ABC in terms of convergence rate and search efficiency, and need more iterations to reach the global optima. The poor effectiveness of FFA may be due to the fact that its parameters are fixed during the search process making it more likely to get trapped in local solutions. ABC shows the best performance in all cases except in f_2, f_4 and f_{12} functions, where GWO is the best performing approach, except in f_{13}, f_{15}. Using the Zakharov problem as an example, GWO manages to reach a minimum value of 3.85E-38, which can be considered to be zero, whereas ABC finds a local solution of 105.886. From the results shown, it is clear that GWO shows a superior performance and robustness in handling high dimensional optimization problems when the number of function evaluations is limited.

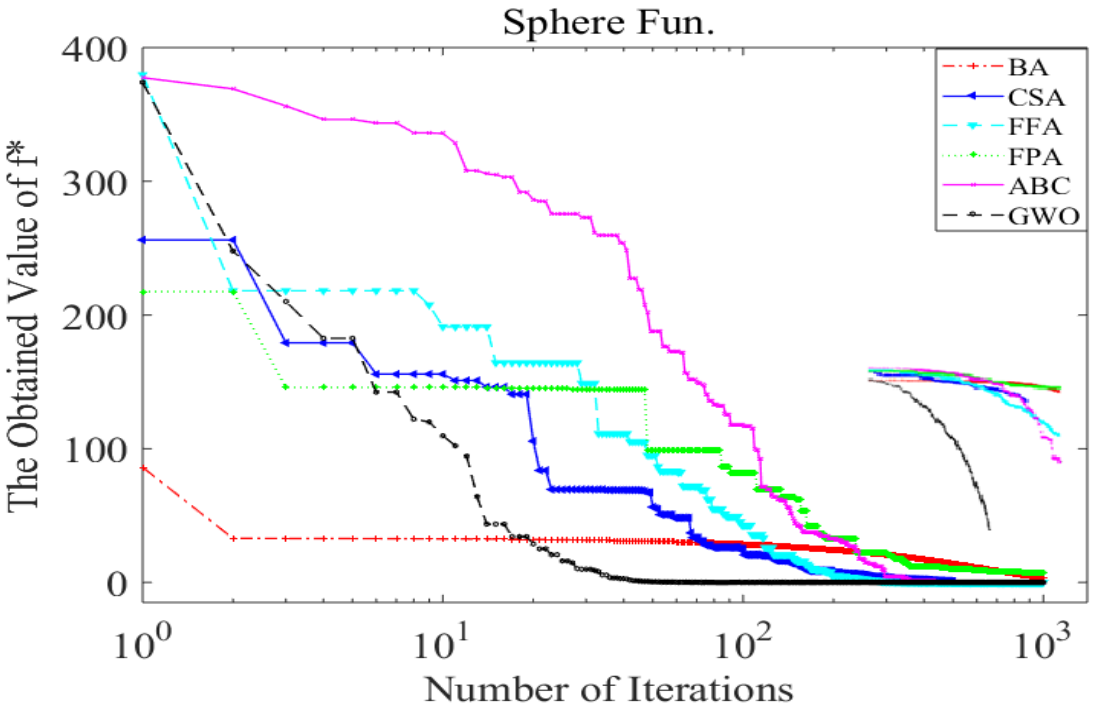


Figure 39.Convergence speed for Sphere (f1) function (50D).

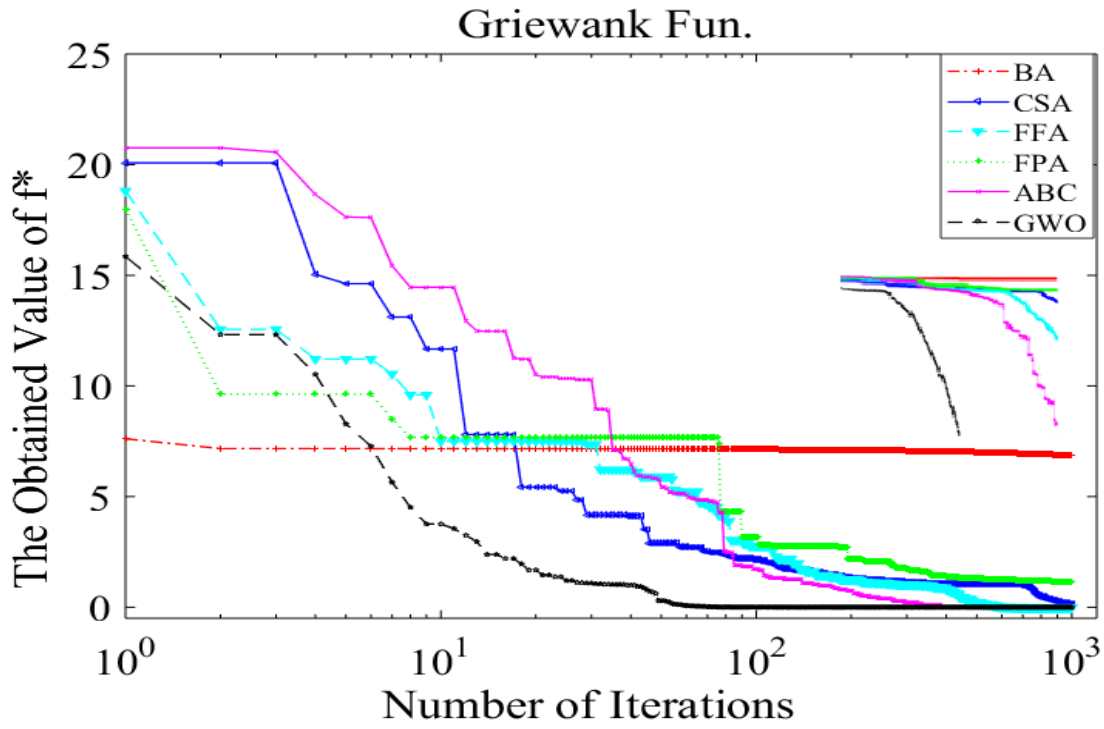


Figure 40.Convergence speed for Griewank (f7) function (30D)

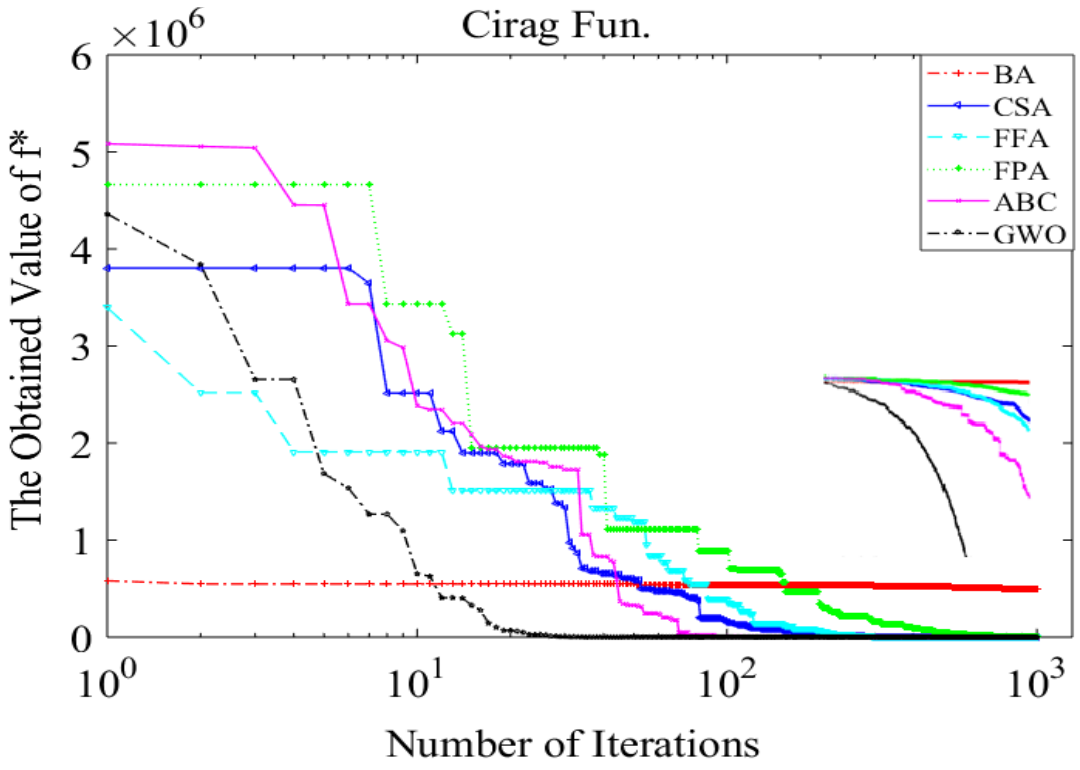


Figure 41.Convergence speed for Cigar (f14) function (25D).

5.7.2 The Computational Complexity Analysis

This sub-section compares these six optimization methods in terms of the required computation time to find the optimum value, using various benchmark functions with different numbers of design variables. The computation time needed to converge is the most significant factor for any method when examining its efficiency. The number of function evaluations (NFE) required by each algorithm on a set of benchmark test functions places the computation time of the six optimization methods under investigation. Figure 42 indicates that GWO and FFA need the lowest and highest average CPU time respectively. As an example, from the statistical results in table 9 for functions f_7 and f_{13}, the FFA method demands a six and seven times higher computation time in comparison to the ABC and GWO approaches. The computational complexity results shown in Figure 5 depict that ABC is more efficient than GWO as its CPU time is almost 50% of GWO for function f_3. ABC shows good efficiency most of the time; however, it can sometimes be stuck into local minima. BA, CSA, FFA and FPA show better performances on low-dimensional problems compared to when it is dealing with high-dimensional GO problems.

The GWO algorithm requires the least computation time, suggesting that GWO is the most promising global optimization method for computationally expensive black-box problems.

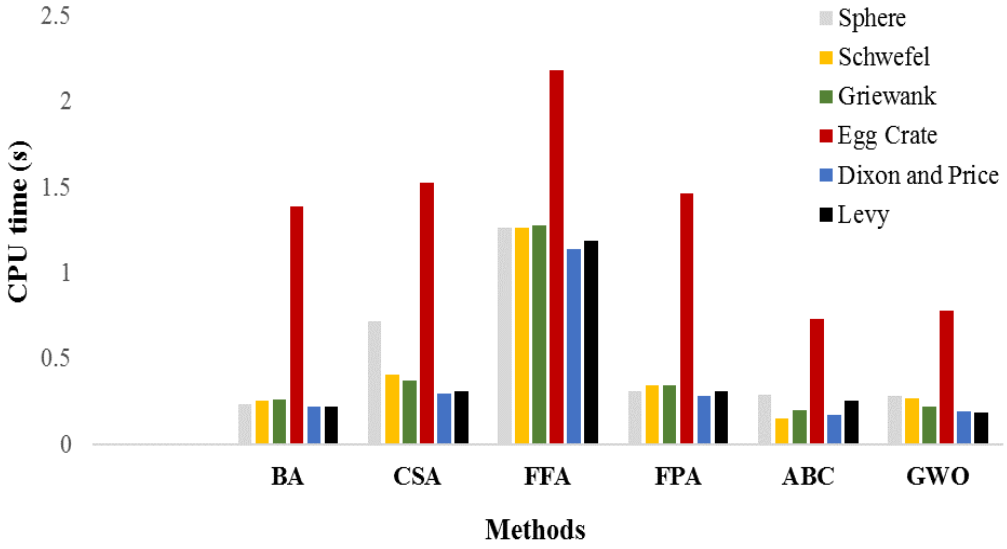


Figure 42. Required CPU time by each method on a set of benchmark function

The influence of increasing the number of variables on the CPU time is presented in Figure 43 Figure 44. As shown, when the number of variables increases, FFA and CSA yield a higher computational complexity compared to the other methods. Furthermore, BA is less sensitive to increasing the number of variables than the other methods while its accuracy is less. Comparing GWO and ABC, to ascertain the best algorithms, shows that GWO needs a relatively lower CPU time, making this algorithm an efficient method for solving many practical GO problems with expensive objective function and constraints.

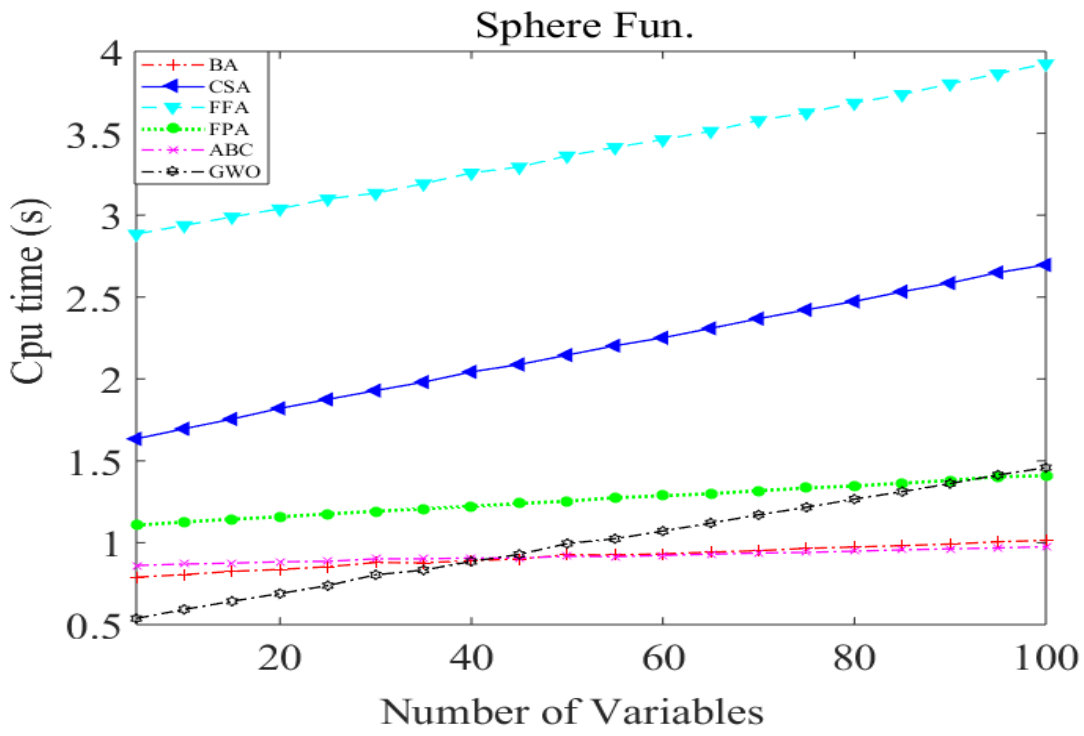


Figure 43. Impact of f dimensions vs. CPU time for Sphere function.

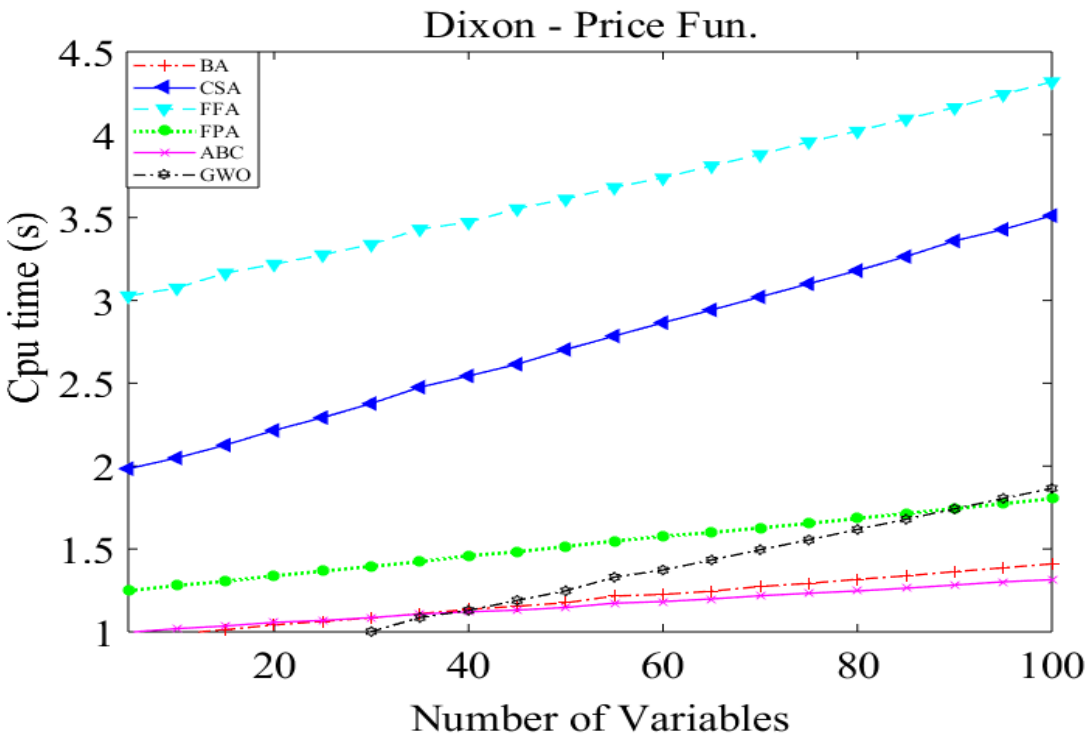


Figure 44. Impact of dimensions vs. CPU time for Dixon and price function.

5.7.3 Impact of Increased Number of Variables on Performance

The aim of this section is to study the impact of high dimensionality on the reliability and accuracy of the solutions achieved by the tested optimization algorithms in solving computationally expensive complex optimization problems. Figure 45 to Figure 47 compare the performance of these six methods for the Sphere, Griewank and Dixon and Price functions. The horizontal axis shows the number of variables, and the vertical axis shows the objective function's error value at a fixed number of function evaluations. Similarly, the value obtained in each iteration serves as a performance indicator. As illustrated, increasing the number of variables leads to the degradation in the performance of BA and FPA. This is owing to the fact that the exploitation capability of BA is comparatively low-especially for high- dimensional problems. The results also reveal that FPA is only efficient in handling low-dimensional expensive black-box functions design problems (less than 10 design variables). On the other hand, GWO and ABC are not as sensitive as the other methods in handling high-dimensional problems regardless of the complexity and the surface topology of the tested problem. The obtained results show that the performance of the GWO and ABC algorithms, which is related to capability and efficiency, is superior for high-dimensional problems. Figure 47 illustrates that the FFA and CS methods continue to provide comparable and acceptable solutions regardless of the complexity of the optimization problem. Furthermore, the results indicate that the performance of these algorithms is roughly similar for low-dimensional unimodal problems ($D < 10$). In summary, the dimensionality strongly affects the performance of most algorithms; however, it seems that ABC is more stable as the dimension of the problems increases which is probably a result of the greater exploration ability of the ABC algorithm.

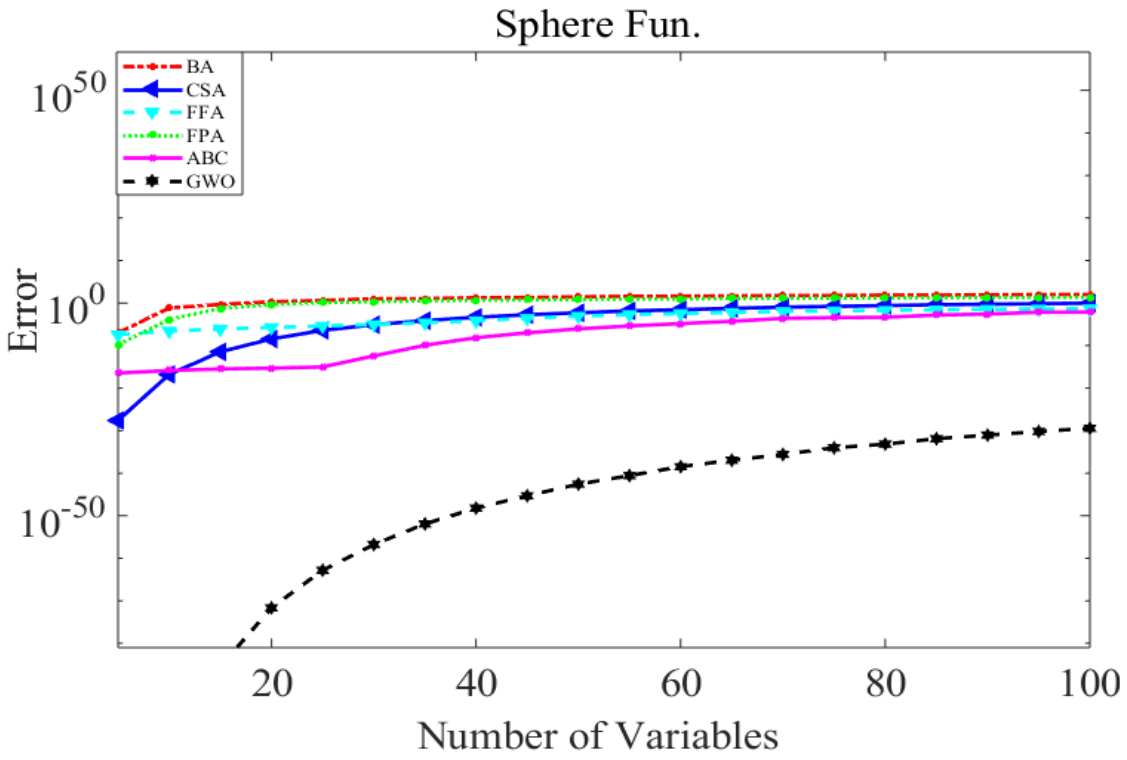


Figure 45. Error vs. variable number for Sphere function.

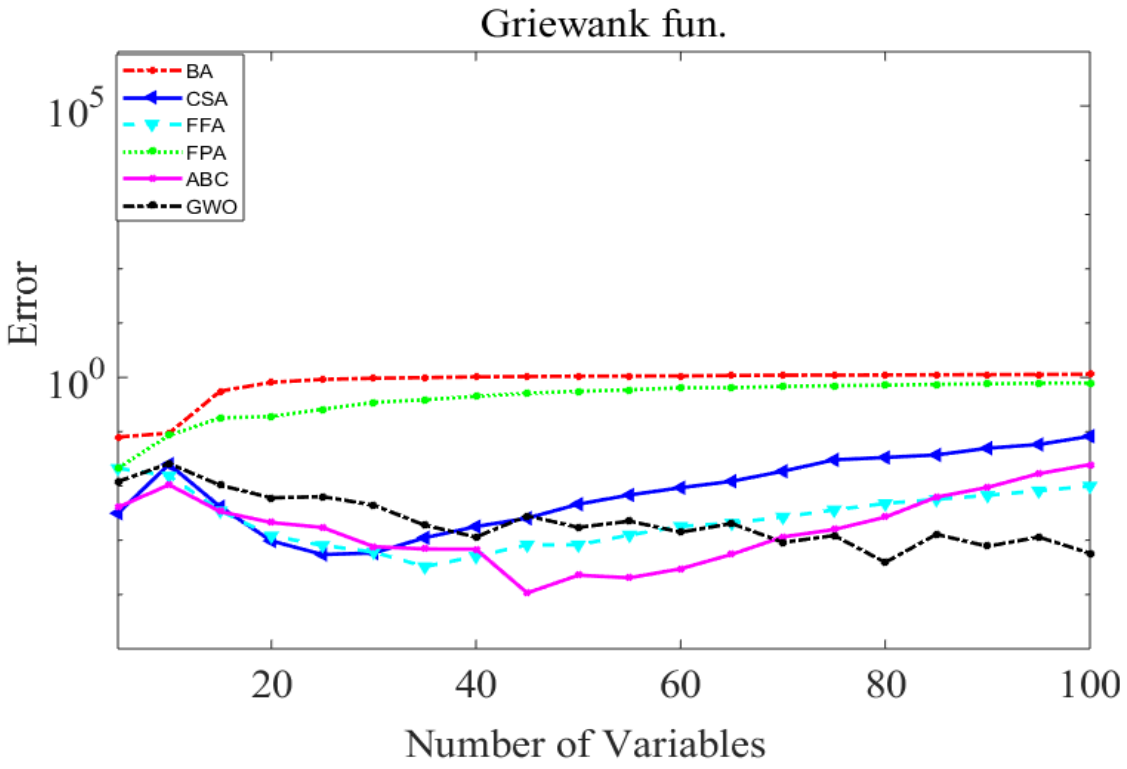


Figure 46. Error vs. variable number for Griewank function

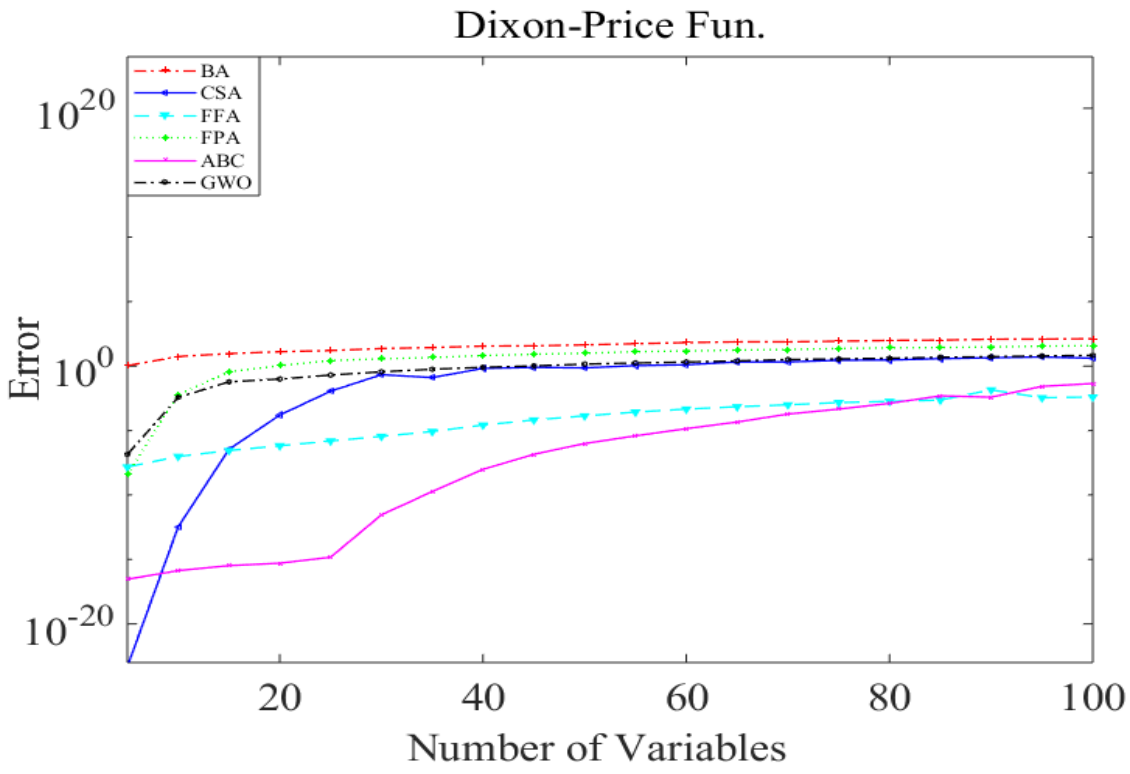


Figure 47. Error vs. variable number for Dixon and Price function

5.7.4 The Overall Performance of the Methods

The overall results obtained in Table 9 can be evaluated based on the properties of the selected benchmark functions. GWO outperforms other algorithms by successfully obtaining the global optimum in 13 out of 15 cases, followed by ABC that outperforms others in 12 out of 15 cases. The third and the fourth best algorithms are FFA and CSA with 9 and 6 respectively out of 15 successful convergence functions, while both BA and FPA solve only 4 of the 15 problems. In terms of accuracy, results suggest that GWO is the best performing approach in all three cases. However, in terms of the required computation time, ABC is superior to the others with an average computation time of 0.2369 seconds, followed by GWO with an average of 0.3136 s. Despite its simplicity, FFA is limited by its tendency to be trapped at local minima, mainly because of its relatively weak global search capability with complex and multimodal problems. That being said, FFA provides excellent solutions in most optimization problems in this work.

In addition, BA shows a poor performance in this study, mainly due to some insufficiencies at exploration and the required high number of function evaluations. Yang [186] BA has been effectively used to handle many challenging GO issues such as multimodal objective problems. Moreover, FPA, known to be a highly efficient method with low-dimensional problems, becomes problematic with high-dimensional problems. Finally, the CS algorithm shows a good performance on number of benchmark tested function. However, the algorithm becomes inefficient when objective function evaluation represents a significant computational cost such as CFD problems. In conclusion, although this study shows GWO and ABC have the highest convergence rates with acceptable computational costs, there is, yet, no universal algorithm capable of obtaining the global solution for all kind of problems. More research needs to be carried out to solve computationally expensive black-box problems. Statistical results in Table 9 presents the overall performance of the tested algorithms under different levels of complexity and dimensionality in order to examine their efficiency and robustness.

5.8 Further Tests Using Nonlinear Constrained Engineering Applications

After testing the methods with fifteen unconstrained high-dimensional benchmark functions, four non-smooth constrained engineering design GO problems were selected to examine the effectiveness of these methods in practice and identify their advantages and drawbacks. These comprised four structural design applications, involving the Welded Beam Design (WBD), the Tension/Compression Spring Design (TSD), the Pressure Vessel Design (PVD) and the Speed Reducer Design (SRD). Several authors in [1, 114, 188-191] have solved these problems using different methods such as the hybrid PSO-GA algorithm, the artificial bee colony, etc. All four of these selected constrained engineering problems have been considered to computationally expensive black-box problems. The problems chosen have design variables ranging from 3 to 7 with 4, 7, 4 and 11 constraints, respectively. An average of 25 independent runs were performed per problem, with a total of 20,000 NEF in each run. All algorithms were tested with the same stopping conditions to ensure that their convergence rate, efficiency and computation time would be examined using the same parameters. Figure 48 shows the coverage history of the Welded Beam

design problem where ABC started the search with lower function values; however, it required additional time to converge, along with the algorithm, to the global minimum. In relation to the tension/compression design problem, Figure 49 shows that GWO outperformed other optimization algorithms, converging to global solution after only a few NFE. In addition, when compared to the other algorithms, GWO converged to global optimal with the least computation time and yielded better minimum values. In the Pressure Vessel problem, FPA had the best performance, with the lowest standard deviation value. FPA yielded the best global optimum function value, taking more computation time to converge in comparison to BA that had the lowest CPU time. Figure 50 compares the required CPU time in dealing with the four selected constrained optimization problems. The obtained statistical results were analyzed and compared to find out the performance of the algorithms when dealing with constrained problems. The results in Table 10 have shown that:

- ABC consistently requires lower NFE than the other algorithms.
- GWO performs well in TSD, WBD and PVD cases, but it seems that the GWO needs more iterations to reach the exact global solution in the SRD case.
- Results demonstrate that the other tested algorithms are sensitive when the number of variables increases, but they can provide competitive search efficiency in some problems.

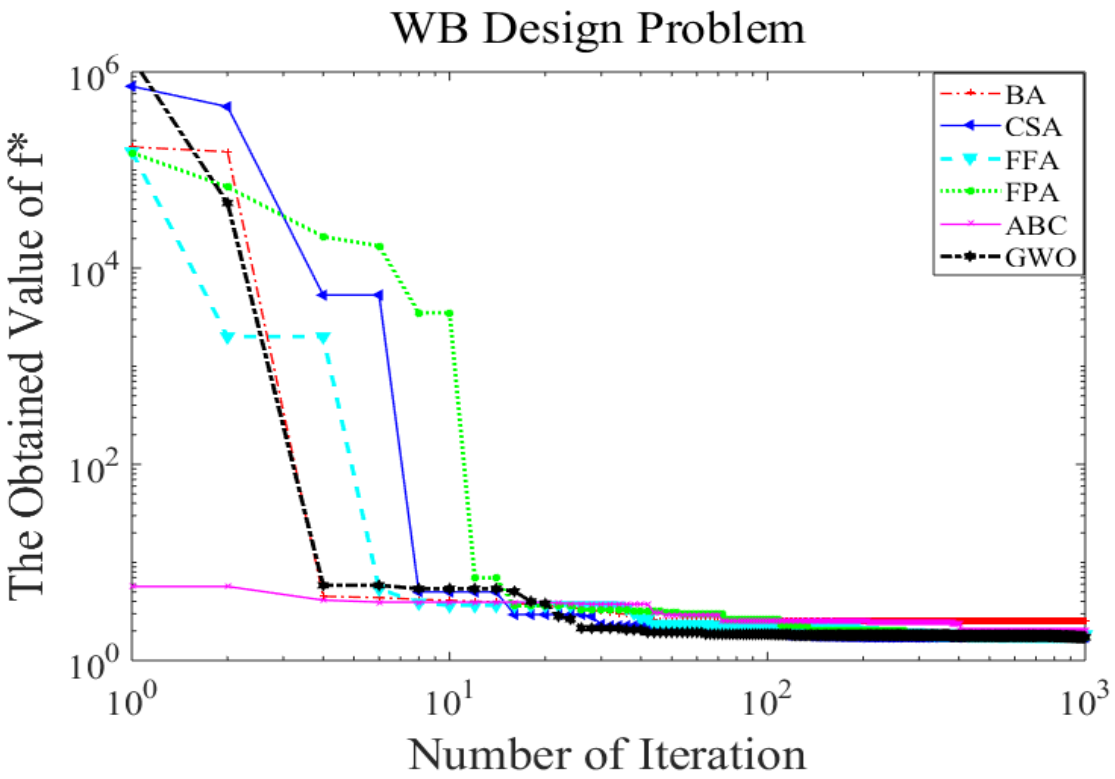


Figure 48. The welded beam problem results

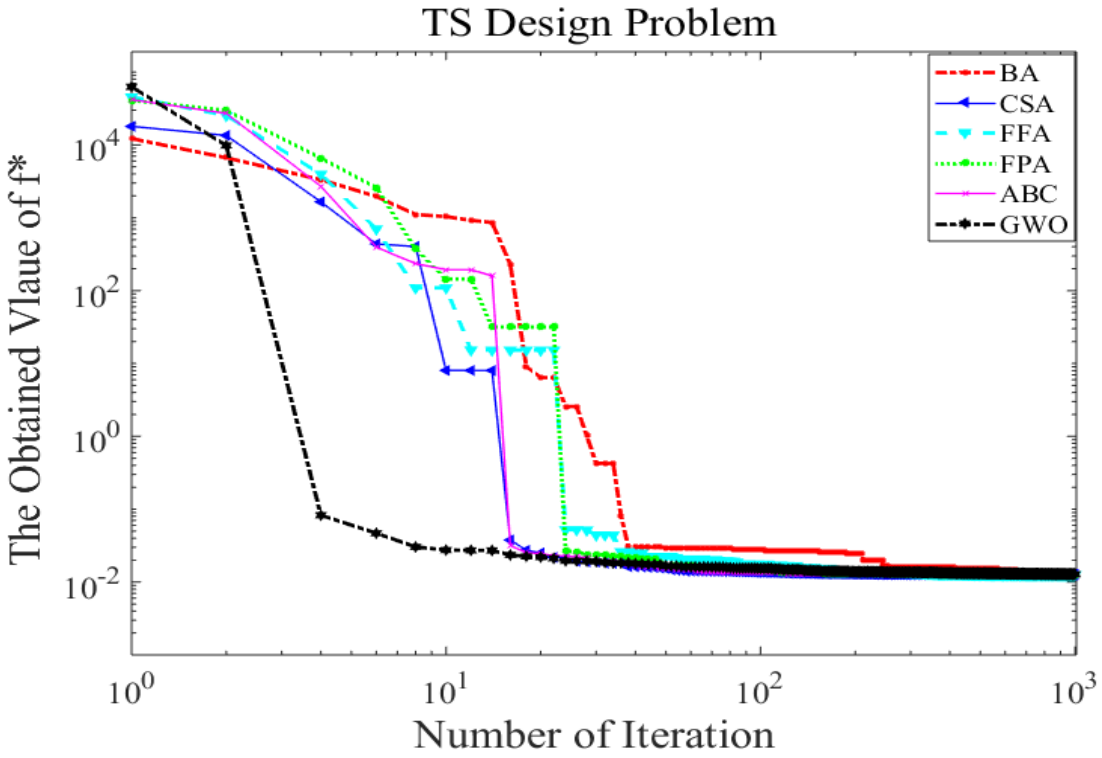


Figure 49. The tension/compression spring problem results

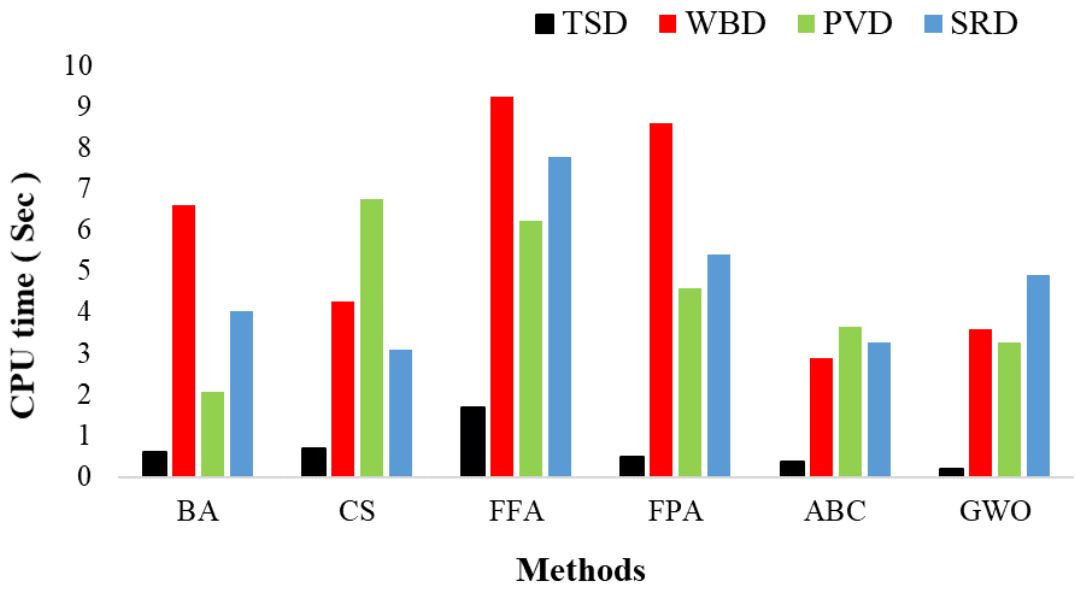


Figure 50. Required CPU time by algorithms for all constrained problems.

The statistical results shown in Table 10, and Figures 48 and 49 showed that no superior optimization algorithm outperforms others for all design problems. Clearly, an algorithm may work well for a specific engineering design problem and fail on another problem. In general, it seems that ABC and GWO optimization algorithms is an extremely promising method for further research and has made significant progress in addressing the analysis and optimization of a variety of complex and expensive systems. From Figure 50 we can claim that the ABC and GWO methods could find the global optimum in lower computation time.

Table 10: Summary of results for nonlinear constrained problems.

Methods	Prob.	Analytical								Obtained f *	NFE	CPU Time
		f *	x_1	x_2	x_3	x_4	x_5	x_6	x_7			
	TSD	0.012665	0.05123	0.34582	13.5571	-				0.013242	20000	0.6078
BA	WBD	1.73852*2*	0.31623	5.5111	6.4044	0.49061	-	-	-	1.8268	20000	6.6091
	PVD	6059.71433	1.04938	0.51773	49.5176	187.5231	-	-	-	6190.4687	20000	2.0781
	SRD	2996.34816	3.5	0.7	17	7.56	7.6892	3.5424	5.2458	3019.5833	20000	4.0369
	TSD	0.012665	0.05178	0.359011	11.1579	-	-	-	-	0.01266	20000	0.6875
CSA	WBD	1.724852	0.20573	3.4705	9.0366	0.20573	-	-	-	1.7449	20000	4.2563
	PVD	6059.71433	0.77714	0.384442	40.44869	178.2513	-	-	-	6074.587	20000	6.7438
	SRD	2996.34816	3.5001	0.7032	17.1	7.4	7.80	3.46	5.27	3000.9597	20000	3.0927
	TSD	0.012665	0.052209	0.370468	10.821	-	-	-	-	0.012713	20000	1.675
FFA	WBD	1.727852	0.19588	3.7176	9.0367	0.20573	-	-	-	1.7291	20000	9.2266
	PVD	6059.71433	0.902631	0.445717	46.92328	177.7547	-	-	-	6026.3068	20000	6.225
	SRD	2996.34816	3.5	0.7	17	7.31	7.820	3.351	5.327	2996.3478	20000	7.771
	TSD	0.012665	0.051643	0.35558	11.3619	-	-	-	-	0.012669	20000	0.4843
FPA	WBD	1.724852	0.20573	3.4705	9.0366	0.20573	-	-	-	1.7568	20000	8.5938
	PVD	6059.71433	0.77447	0.38323	40.32289	179.959	-	-	-	5990.4928	20000	4.5969
	SRD	2996.34816	3.5	0.761	17.001	7.1	7.54	3.3521	5.3412	3019.5833	20000	5.4109
	TSD	0.012665	0.052855	0.37997	10.1552	-	-	-	-	0.012657	16092	0.3796
ABC	WBD	1.729852	0.20866	3.3293	8.9139	0.20453	-	-	-	1.7251	17145	6.8906
	PVD	6059.71433	0.84036	0.426735	42.56989	176.4665	-	-	-	6059.037	16200	3.6609
	SRD	2996.34816	3.512	0.7	17	7.3	7.8	3.3502	5.2866	2996.3088	18700	3.2701
	TSD	0.012665	0.051149	0.305127	10.7273	-	-	-	-	0.012665	20000	0.2031
GWO	WBD	1.724852	0.20431	3.5018	9.0378	0.20576	-	-	-	1.7266	20000	6.5938
	PVD	6059.71433	0.8125	0.434511	42.0891	176.7587	-	-	-	6059.7639	20000	3.2563
	SRD	2996.34816	3.46427	0.7129	17.1001	7.25581	7.18738	3.2359	5.7814	3005.6092	20000	4.9021

5.9 Floating Offshore Wind Turbine Support Structures Cost Minimization

In this subsection, a real industrial application of global optimization on the production cost minimization of a Floating Offshore Wind Turbine Support Structure (FOWT) is used to illustrate and test the newly introduced algorithm. Wind energy technology is growing rapidly at commercial scales and is one the mature means of renewable energy generations [192]. At the coastlines, higher wind speed resources are available and they provide consistent wind power. Therefore, offshore wind turbine technologies require floating platforms to harvest the wind energy in deep waters instead of using fixed structures which are suitable for shallow waters [193]. This section provides a real-life application for a group of optimization algorithms to evaluate FOWTs support structure cost based on the method reported in [193]. A spar buoy platform is one of the structures that is currently used in offshore wind projects and has been selected in this study for cost analysis as shown in Figure 51.

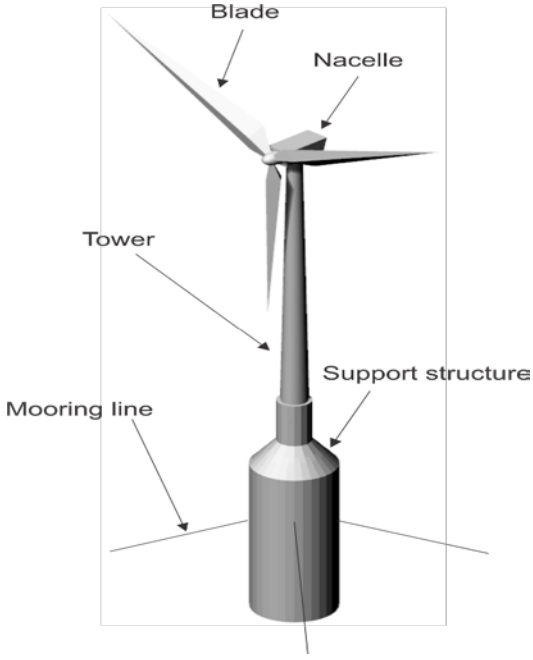


Figure 51. (FOWT) with a spar buoy support structure

The approach used in this research is to define a design parameterization for the support structure as well as a linearized frequency domain dynamic model which evaluates the

stability of the floating system using the NREL 5 MW offshore wind turbine [193, 194]. The support structure parametrization pattern used in this work provided the spar buoy platforms with a limited number of design variables. Hence, four design parameters are defined to shape the platform and mooring system: a central cylinder with a draft, HI, cylinder radius, RI, top tapper ratio, TI, and a catenary mooring system, XM for spar buoy platforms. Note that a free board (FB) of 5 m is used for all the spar buoy platform designs as a constant parameter. The geometric design variables are summarized in Table 11 and Figure 52. In this work, the mooring system design variable, XM defines the angled taut line and slack line configurations for the spar buoy platform. Three lines are used for this platform and they are located at the middle the cylinder height as presented in Figure 52. The anchor position is defined by the mooring variation of XM from positioning under the platform to the horizontal spread of the lines. The mass properties for each spar buoy platform are determined using the platform and mooring design parameters. The mass of support structure is calculated using a wall thickness of 50 mm using a steel density of 8050 kg/m^3.

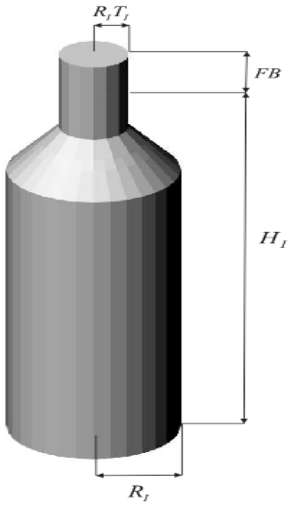


Figure 52. Design characteristics of a spar buoy platform

Table 11: Geometric design variables of spar buoy platform.

Variable	Description	Min.	Max.
HI	*Central cylinder draft*	*2 m*	*150 m*
RI	*Central cylinder radius*	*3 m*	*25 m*
TI	*Top tapper ratio*	*0.2*	*2*
XM	*Mooring System*	*0.2*	*2*

In this study, the cost of the FOWT design is defined based on three main components that include the mooring line, the 5 MW wind turbine, and the floating platform. To determine the stability of the floating system in a given environmental condition, the linearized characteristics of the hydrodynamics and aerodynamics of the FOWT must be collected in a frequency domain dynamic model. For floating platform hydrodynamics, the added mass, damping, and hydrostatic loads are calculated using WAMIT. The wind turbine dynamic characteristics are kept constant in this study. However, FAST is used to calculate the linearized dynamic properties for the NREL 5 MW offshore wind turbine. These properties include linearized stiffness, mass, and damping matrices of the offshore wind turbine in a given wind speed. Like the floating platform hydrodynamics, linearized mooring line properties can be added to the frequency domain dynamic model using a quasi-static mooring subroutine of FAST. All the calculated coefficients and loads are collected into a 6 × 6 system stiffness, mass, damping to generate the linearized frequency domain equation of motion for a FOWT Equation (5.1). The stability of the system is evaluated by calculating the complex response amplitude operator (RAO) for all modes of motion. More details about the dynamic model and platform parameterization can be found in [194].

$$-\omega^2 M_{total}(\omega)\hat{Z}(\omega) + i\omega B_{total}(\omega,\zeta)\hat{Z}(\omega) + C_{total}\hat{Z}(\omega) = \hat{X}(\omega) \tag{5.1}$$

$$[RAO_1(\omega) :: RAO_6(\omega)] = [-\omega^2 M_{total}(\omega) + i\omega B_{total}(\omega,\zeta) + C_{total}]^{-1}\hat{X}(\omega) \tag{5.2}$$

The performance of the FOWT in this study is defined as the standard deviation of nacelle acceleration and is shown below:

$$\sigma_{anac}(\omega) = \sqrt{\int_0^\infty |RAO_{anac}(\omega)|^2 \; S(\omega)\, d\omega} \tag{5.3}$$

Where, $S(\omega)$ is the wave spectral density, and $RAO_{anac}(\omega)$ is the fore-aft nacelle acceleration response amplitude operator. The complex forms of the nacelle displacement RAO and nacelle acceleration RAO are given by Equations (5.4) and (5.5):

$$RAO_{\zeta nac}(\omega) = (RAO_1(\omega) + RAO_5(\omega)\, z_{nac}) \tag{5.4}$$

$$RAO_{a_{nac}}(\omega) = -\omega^2(RAO_1(\omega) + RAO_5(\omega)z_{nac}) \quad (5.5)$$

Where z_{nac} is the wind turbine hub height. In Eq. (5.4) and (5.5) the $RAO_{anac}(\omega)$ includes the aero dynamic effects of the wind turbine implicitly from the linearized dynamic properties of the wind turbine. $Z_{nac}RAO_{anac}$ The objective function to optimize in this study is the support structure cost including mooring system, anchor and floating platform costs:

$$min\ f(x): C(x) = C_{Platform}(x) + C_{Mooring}(x) + C_{Anchor}(x) \quad (5.6)$$

The floating platform cost is defined as a function of the design parameters as shown in Table 11. In this study, the platform cost contains the materials, manufacturing, and installation costs using a constant cost factor of \$2.5/kg [193, 194]. The mooring system cost is calculated using the length of the mooring lines and the maximum steady state mooring line tension with a factor of \$0.42/m kN. The anchor cost is the third component of the cost model that includes the installation and technology costs. The drag embedment anchor technology is selected in this study for catenary mooring system. Therefore, the installation cost for each anchor is defined as 100 \$/anchor/kN and the technology cost is based on 5000 \$/anchor.

In this study, a number of constraints are also applied to the objective function. To avoid expensive design configurations, a constraint limits the cost of the platform to less than \$9 M. The performance metric $RAO_{anac}(\omega)$ is also limited to less than 1. In addition, to avoid the floating platform turning over, the steady state pitch angle of the platform should be less than 10°. The expression of this constraint is given in the following equation:

$$\zeta_5 = \frac{F_{thrust}.z_{nac} + M_{mooring_5} - M_{ballast}}{\rho g \forall_{ZCB} - M_t g z_{CG} + \rho g\, I_{xx} - C_{mooring5,5} + C_{5,1} z_{fair}} \quad (5.7)$$

where, F_{thrust} is the steady thrust load, $M_{mooring_5}$ is the mooring line pitching moment, $M_{ballast}$ is the pitching moment from the ballast mass stabilizers, $\forall$ is the platform displacement, z_{CB} is the center of buoyancy of the platform, M_t is the total mass of the FOWT, z_{CG} is the center of gravity location, I_{xx} is the water plane moment of inertia in platform pitch motion, $C_{mooring5,5}$ is the mooring lines stiffness in pitch motion, $C_{5,1}$ is

the mooring line stiffness in pitch-surge motions, and z_{fair} is the fairlead depth in pitch motion. For details of each constraint, the reader is referred to the study presented in [194]. The optimization methods have been tested on this real-life engineering optimization problem. The exploration space for the various platform concepts helps identify the effective cost of structure for FOWTs. In order to obtain solid results, 25 independent runs have been performed. The test results are given in Table 12 and the convergence history of each method is illustrated in Figure 53. The ABC method still provides outstanding performance and could find the global optimum within 951 function evaluations of the original functions in this case study. According to the results from previously published performance tests [194], the obtained optima are sufficiently accurate and the recorded NFEs are much lower. Figure 54 compares the optimization methods in terms of the required CPU time.

Table 12: Summary of comparison result for the cost of FOWTs application.

Methods	***f** Min**	**NFE**	**CPU Time**
BA	*5533111.9378*	*1000*	*121.1719*
CSA	*4426006.2921*	*1000*	*123.5938*
FFA	*4811655.8121*	*1000*	*131.8125*
FPA	*4727085.732*	*1000*	*132.875*
ABC	*3817499.5279*	*951*	*119.2969*
GWO	*4395036.2793*	*1000*	*123.5313*

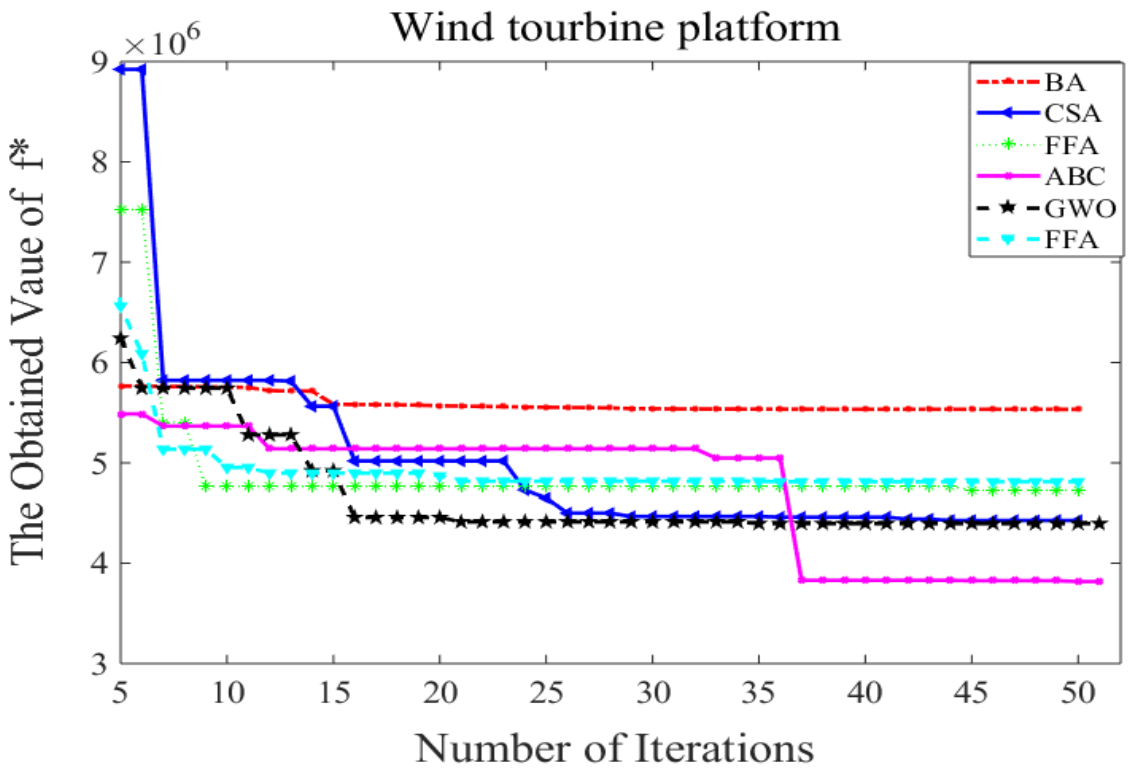


Figure 53.Cost for FOWT problem results

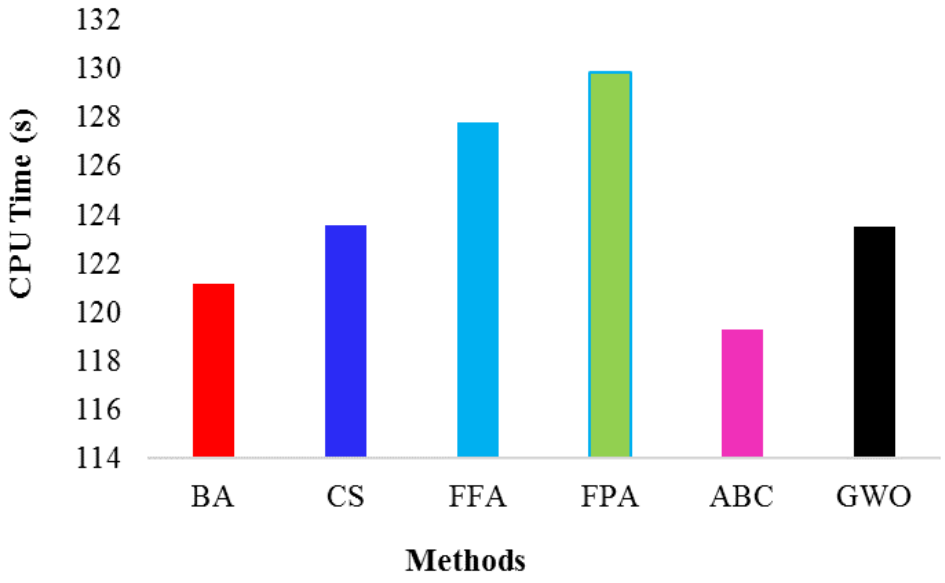


Figure 54. Required CPU time for FOWT problem.

5.10 Summary

New and advanced GO algorithms have been continuously introduced to solve complex design optimization problems. These problems are characterized by using computation intensive numerical analyses and simulations as objective and constraint functions with a large number of design variables, forming the so- called computation intensive, black-box GO problems. The relatively new, nature-inspired global optimization methods, Artificial Bee Colony (ABC), Firefly Algorithm (FFA), Cuckoo Search (CS), Bat Algorithm (BA), Flower Pollination Algorithm (FPA) and Grey Wolf Optimizer (GWO), have been introduced to improve search efficiency by reducing the needed number of objective function evaluations and increasing the robustness and accuracy of the results. This study is aimed at gaining a better understanding of the relative overall performance of these six well-known new GO search methods. The systematically conducted comparative study under the same evaluation criteria has quantitatively illustrated each algorithm's ability to handle a variety of GO problems to enable a user to select an appropriate GO tool for their specific design applications. Fifteen high-dimensional, unconstrained benchmark functions with unimodal and multimodal shapes; four constrained engineering optimization problems; and a real-life floating offshore wind turbine design optimization problem have been used to conduct the performance and robustness tests on these newer GO methods. These selected complex benchmark functions and real-life computationally expensive design optimization examples present either convergence or computation challenges.

The GWO and the ABC algorithms have been found most capable in providing the optimal solutions to the high-dimensional GO problems. The main advantages of the GWO method include quick convergence, more accurate results, highly robust, and strong exploration ability over the search space with an overall efficiency of 86%. The ABC method ranks as the second best with an overall efficiency of 80%. The FFA, CS, PFA and BA methods were found to be able to provide competitive results with other well-known algorithms, including SA, GA and PSO. Results from this comparative study show that the ability of these GO methods to obtain a good solution also declines as the dimension of the problem increases, and each specific application requires a particular type of GO search algorithm. The research contributes to future improvements of global optimization methods.

Chapter 6. Extension of Bat Algorithm Using Kriging SM

6.1 Introduction

Today's increasingly competitive environment calls for design optimization of variousintricate industrial products that often present as a complex multiphysics system. The functional performance evaluations of these systems frequently require CEBB simulations and analyses. Many challenges arise in solving these GO problems, including complexity, differentiability, dimensionality, and computation intensity. Since the early 70's, many stochastic and nature-based GO algorithms have been introduced as dedicated methods for optimization problems with non-unimodal, discontinuous, discrete, and ill-shaped functions, from the earlier Genetic Algorithm (GA) to the more recent approach of BA. These methods have been proven to be efficient in solving many difficult and challenging GO problems [195, 196].

On the other hand, many these GO methods require a large number of statistical sample points that are difficult or "expensive" to obtain from the CEBB objective functions, leading to slow convergence and long, if not impossible, computation time. More research has been devoted to improving the performance of these GO algorithms. Yang [151] introduced one of the more recent GO methods, called BA, based on population of solutions, which simulates the behavior of bats searching for prey, avoiding obstacles, and finding their way in the dark using their echolocation system. BA has proved to be an effective algorithm to locate the global optimum and finds applications in solving various challenging optimization problems, including motor wheel optimization[197], and multi-objective optimization BA (MOBA) [198]. [199] used BA for solving hybrid flow shop scheduling problems, and Luo et al. [200] developed a discrete BA for addressing the optimal permutation flow shop scheduling problem. Their results indicate that BA is an effective method for solving such problems. Considerable efforts have been devoted to further improve the exploration and exploitation capabilities of the BA and to test the algorithm in various engineering optimization applications [201]. Fister et al. [202] introduced a new version of the BA to overcome the insufficiency and correct the planning of sports training. Gandomi and Yang [203] modified the original BA method by introducing chaos into BA called (CBAs) to improve its search efficiency. Wang and Guo

[157] also hybridized the BA with the Harmony Search Algorithm for global numerical optimisation problems. Nakamura et al. [204] presented a revised BA that combines the exploration power of the bats with the speed of the optimum-Path Forest classifier. Li and Zhou [205] modified the BA using complex-value to increase the diversity of the population and improve the exploration capability. Roeva and Fidanova [206] presented a new hybrid method, which combines the BA with sequential quadratic programming for facing the parameter identification of an E. coli fed-batch cultivation process model. Hasancebi et al. [207] developed a novel BA that employs fundamental principles of the BA search to solve structural optimization problems. A year later, Hasancebiand Azad [208] also modified BA to solve discrete size optimization problems for steel frames and compared the results with other GO algorithms. Lin et al. [155] presented a combination of BA with Levy flights and chaotic maps for parameter estimation of dynamic biological systems. Gandomi et al. [209] used BA to solve both constrained and unconstrained benchmark problems, and real-world optimization problems. Yang et al. [182] compared the efficiency of the BA with intermittent search approaches. Peres et al. [210] compared BA with other nature-based approaches for solving power system optimization problems. Sathya and Ansari [211] used BA for solving controller optimization design problems. Furthermore, Ramesh [212] used BA for multi-objective optimization problems in power system. Lastly, Osaba et al. [213] presented discrete Bat Algorithm for solving the Traveling Salesman Problem and the Asymmetric Traveling Salesman Problem. Although all of these efforts have modified the BA for their specific applications, none has directly addressed the specific need of the CEBB GO problems. In this work, improvements of the BA have been made by introducing the Kriging surrogate model (SM) to address the specific need of reducing the number of objective function evaluations (NFE) in solving CEBB GO problems.

6.2 The Bat Algorithm (BA)

As a nature-1nspired, population-based algorithm (BA) [151], is a random search swarm intelligent algorithm that mimics the behavior of micro-bats, using their echolocation system to identify prey, avoid obstacles, and find their way. In BA, the location of each bat represents a candidate solution of the optimization problem. The location of the prey found by the i_{th} bat can be expressed as $x_i = (x_1, x_2 ... x_D)$, as potential improvements. The fitness value of x_i represents to the quality of the location. The algorithm uses the following concepts:

- Bats use their echolocation system to measure distance and identify the difference between thesource of prey and obstacles.
- Bats fly randomly with velocity, V_i, at location, x_i, emitted pulses at frequency, f_i, wavelength, λ, and loudness, A_0, searching for food. They change the frequency of the pulse automatically and regulate the rate of its emission, $r \in [0, 1]$, based on the distance to their target.
- The loudness of the emitted pulses varies between the minimum $A_{\min}$ to A_0.

Each bat will fly to an updated best position, and, at each iteration, its location, velocity, and frequency are adjusted by:

$$\begin{aligned} f_i &= f_{min} + \beta(f_{max} - f_{min}) \\ v_i^{t+1} &= v_i^t + (x_i^t - x^*) f_i \\ x_i^{t+1} &= x_i^t + v_i^{t+1} \end{aligned} \tag{6.1}$$

where f_i is the pulse frequency emitted by each bat, f_{min} & f_{max} are minimum and maximum frequencies, β is a randomly generated vector, x^* is the best global location obtained so far after comparing all solutions between N bats, and v_i^t is the bat's velocity at step time, t. When the best solution is found among the current bat locations, the BA searches locally near the obtained solution to generated a new candidate:

$$x_{new} = x_{old} + \varepsilon A^t \tag{6.2}$$

Where, $\varepsilon \in [-1\ 1]$ is a random number, A^t is the average of loudness value of all the bats at time step *t*, and x_{old} is the present best solution.

The loudness, ***A*,** and pulse emission rate, ***r*,** are updated as the bats find prey. ***A*** will decreases while ***r*** will increases gradually as the bat approaches the prey.

$$A_i^{t+1} = \alpha A_i^t \tag{6.3}$$
$$r_i^{t+1} = r_i^0(1 - e\gamma^t)$$

Where, α and γ are both constants. α controls the convergence rate of BA and plays a similar role as the cooling factor in the simulated annealing (SA). For simplicity, we set = γ = 0.9 in our simulations. The BA search algorithm consists of the following steps:

Bat Algorithm

1: Objective function $f(x)$, x $\in (x_1, \ldots\ldots\ldots, x_n)^T$
2: Generate the bat population xi and vi for i = 1.....*n*
3: Define pulse frequency $f_i \in [fmin, fmax]$
4: Initialize pulse rates r_i, and the loudness A_i
5: **while** (k < Max of iterations) or (*fmin* > *tol*)
6: Generate new solutions by adjusting frequency, and updating velocities and locations for *i*
7: **If** (rand (0,1) > r_i)
8: Rank the solutions and choose the best one
9: Define a local solution around the best solution selected
10: **End if**
11: Generate a new solution by flying randomly
12: **If** (rand(0,1) <A_i and $f(x_i) < f(x^*)$) then
13: accept the new solutions
14: adjust r_i and A_i
15: End **If**
16: Rank all the solution and choose the best one
17: Update iteration number k = k+1
18: End while
19: The final outputs are x_i^*
End

6.3 Surrogate Models (SM)

A surrogate is a mathematical approximation method to predict the behavior of a system using a set of sampling points, normally acquired from computation intensive simulations. In the approach, a set of design variables as inputs to the surrogate model, produce system outputs or responses. Many surrogate modeling methods have been introduced for various

applications, including the response surface method (RSM) or polynomial regression, Kriging, radial basis functions (RBF), support vector regression (SVR). a comprehensive Reviews on the techniques have been made in [72]. Design of experiments (DOE) is the technique for placing the test points properly within the design space to effectively estimate the actual system model, using one of the surrogate techniques. A number of DOE techniques have been widely used such as fractional factorial, central composite design (CCD), face-centered design, Box-Behnken and Latin Hypercube Sampling (LHS) [214] as sampling methods in the search space . As the most well-known space sampling method with outstanding performance [104, 215] , the LHS is used in this work to generate sample points over the search space for building the Kriging model.

Kriging is a model prediction method introduced by Daniel Krige [214], which interpolates experimental data at all sample points by minimizing its Mean Squared Error (MSE). Kriging is useful in a wide range of engineering design applications as a global approximation technique and is particularly suitable for high dimensional non-linear and multimodal problems [104]. The Kriging predictor and its estimated MSE function have the following forms:

$$\hat{f}(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{f} - \mathbf{1}\hat{\mu}) \tag{6.4}$$

$$\hat{s}^2(\boldsymbol{x}) = \hat{\sigma}^2\left[1 - \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}) + \frac{(1 - \mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}))^2}{\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\mathbf{1}}\right] \tag{6.5}$$

where, $\hat{f}(x)$ is the approximation of the function *f (x),* $\hat{\mu}$ is the result by simply plugging x_i into the regression equation, and the second term of Eq. (4) represents the adjustment to this prediction based on the correlation of $x_i \in R^p$ with the error terms at the sampled points. If there is no correlation ($r=0$), then $\hat{f}(x) = \hat{\mu}$. In the Eq. (6.4) $\hat{s}^2$ reflects the uncertainty of the predictor. If the to-be-tested location $\boldsymbol{x}$ is far away from the known points, it will have a higher value, indicating a large prediction error. The term of $-r(x)^T\mathbf{R}^{-1}\boldsymbol{r}(\boldsymbol{x})$ represents the reduction in prediction error due to the fact that x is correlated with the sampled points. The second part $(1 - \mathbf{1}^T\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}))^2 / 1^T\boldsymbol{R}^{-1}\mathbf{1}$ reflects the uncertainty that stems from not knowing μ exactly. A full derivation of this relation can be found in Appendix B. Kriging is one of the most common methods for such a stochastic model interpolation. Kriging is particularly flexible due to the wide range of correlation

functions R (u; v), which can be selected. In this work, Ordinary Kriging is adopted due to its simple structure, ease of implementation and fast execution. Other Kriging methods with more complex form, including Universal and Blind Kriging, rely on priori trend information from an application, which is difficult to obtain. More details on the steps of construction of Kriging SM are in Appendix B.

The Kriging model prediction process is illustrated here using the 2D Banana unimodal function as shown in Figure 55 (a-c). Twenty training points are used to construct the initial Kriging model, as shown in Figure 55(b), to capture the rough shape of the original function. For a high dimensional problem, more sample points will be needed. As the sample points increase the predicted shape closely approximates the shape of the original function as shown in Figure 55(c).

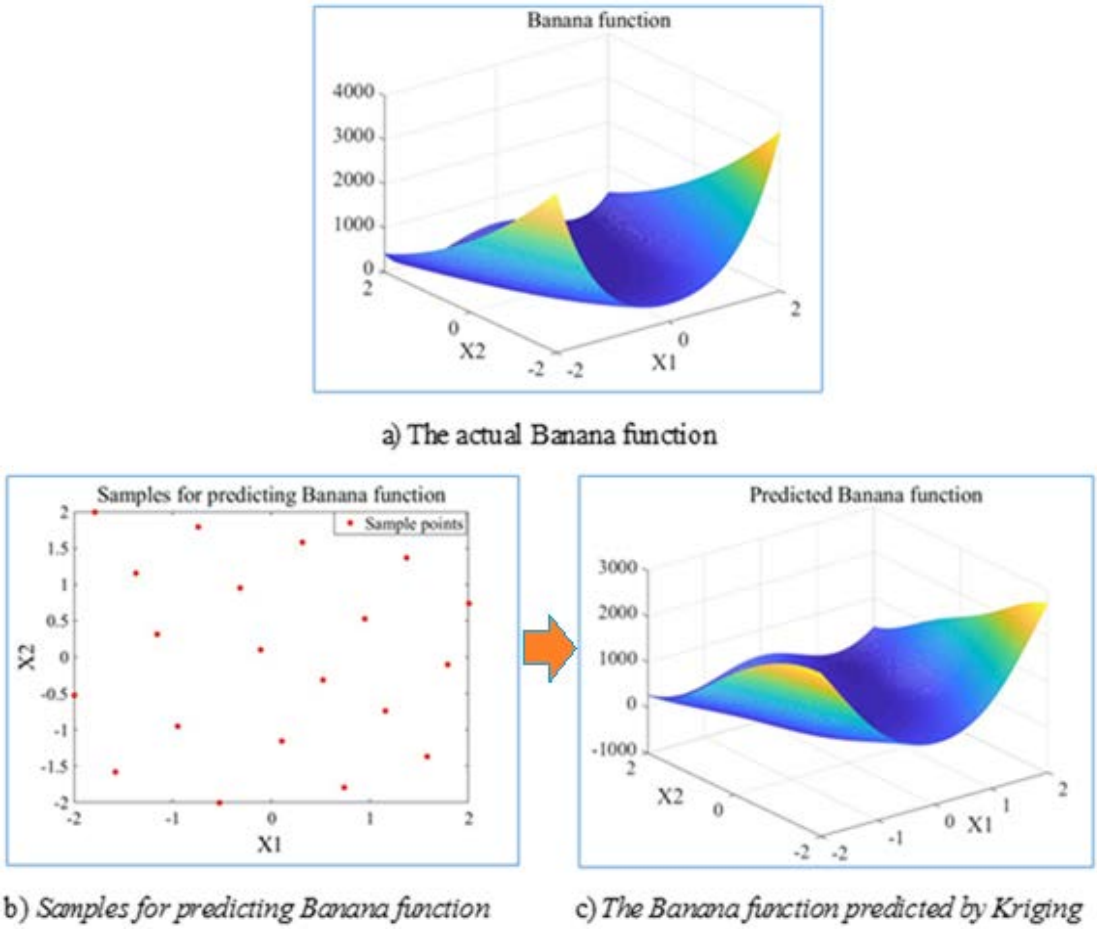
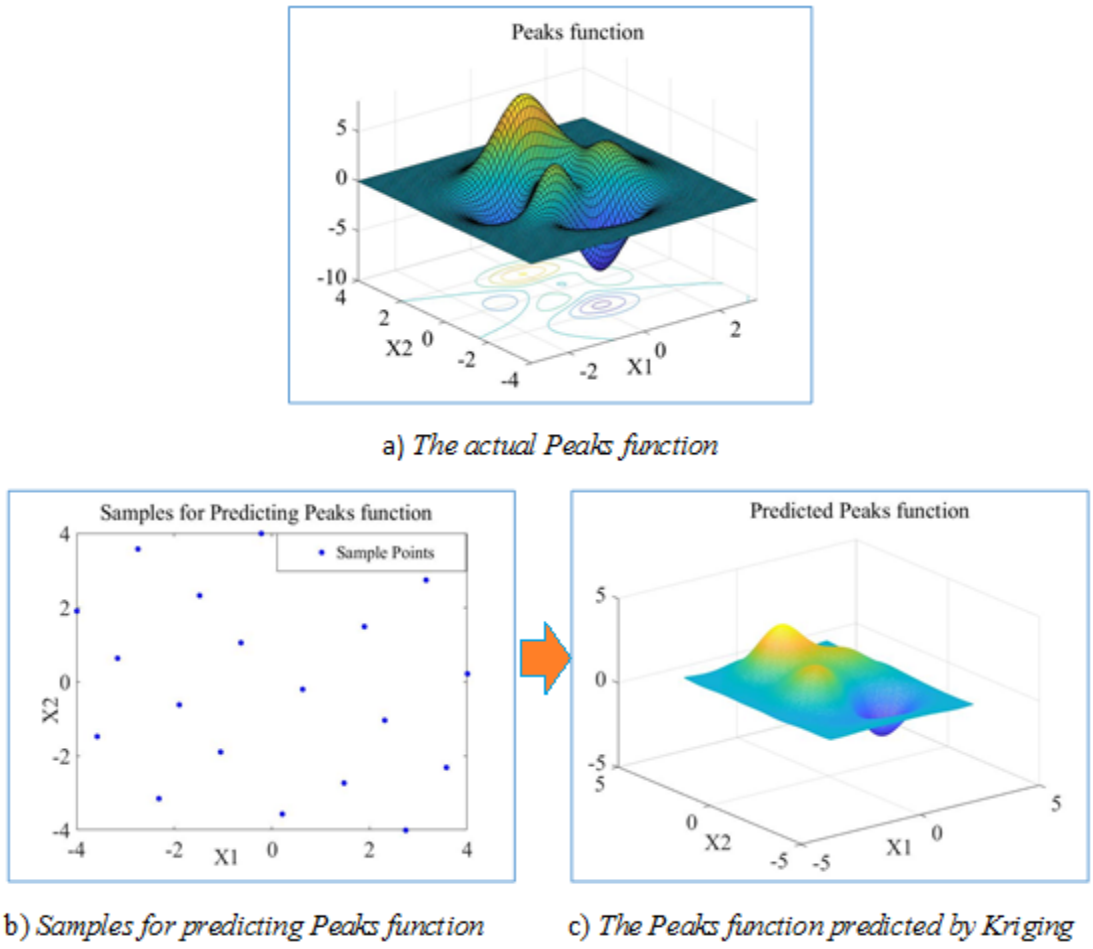


Figure 55. Kriging prediction process on unimodal Banana function.

The Figure 56 (a-c) illustrations the Kriging estimation process of the multimodal Peaks function. Twenty sample points were used in the design space using the LHS to estimate the Peaks function with a reasonably accurate approximation. A study that compared different metamodeling techniques concluded that Kriging has smaller modeling errors and serves as the best surrogate predictor.



a) *The actual Peaks function*

b) *Samples for predicting Peaks function*

c) *The Peaks function predicted by Kriging*

Figure 56. Kriging prediction process on multimodal Peaks function.

6.4 Kriging-SM Assisted Bat Global Optimization Algorithm

The newly proposed K-BA GO search method consists of two parts: (a) a Kriging model was employed to construct an approximation model of the objective function during the evolution process; and (b) BA was used to optimize the approximated model. The algorithm consists of the following steps, as illustrated in Figure 57.

Steps of the Proposed Algorithm:

Step 1: Generate a set of design data points, $x_i, i = 1, \dots, n$ in the whole search space using the LHS method based on the following formula:$x_i = \{x_1, x_2, \dots, x_n\}, \ \ x_i \in S^p$.

Step 2: Calculate the fitness of the expensive function using the selected points, save, and rank them into database.

Step 3: Construct the Kriging-based SM approximation model over the design space based on the expensive point in database.

Step 4: Apply BA to find a group of solutions under Kriging surrogate model and evaluate their expensive function values. This group of solutions is the final generation of BA on Kriging.

Step 5: If the number of iterations is the multiple of 4, BA is used to maximize the estimated MSE of Kriging. The solution with the maximum MSE value will be chosen as a supplementary point.

Step 6: Add the newly generated points into the previous expensive sample set. Delete the repeated points.

Step 7: Calculate the fitness of the expensive function and rank all the expensive samples.

Step 8: If the best value satisfies the stopping criteria, terminate the loop. Otherwise, repeat steps (3) to (7) until the stopping criteria is satisfied.

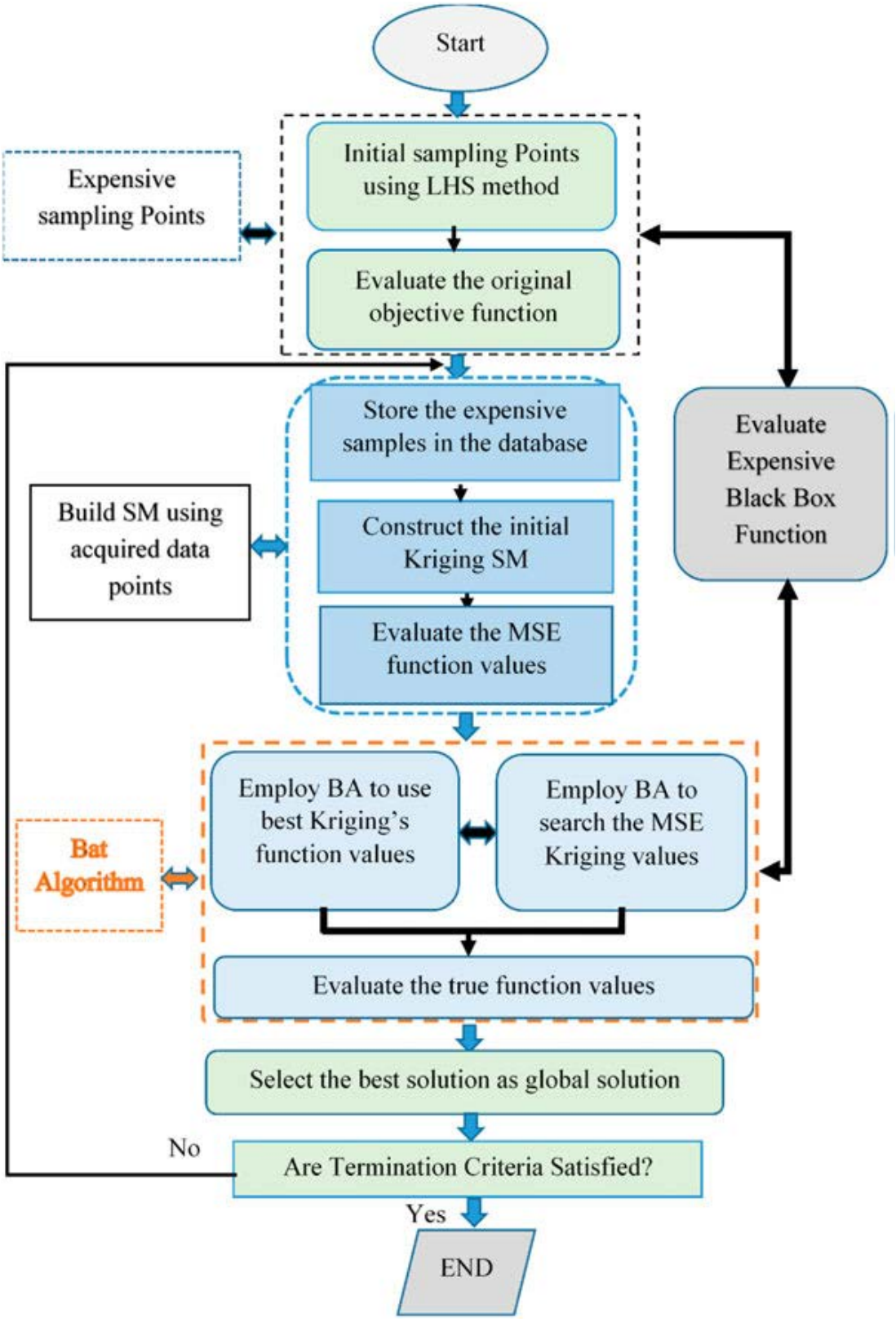
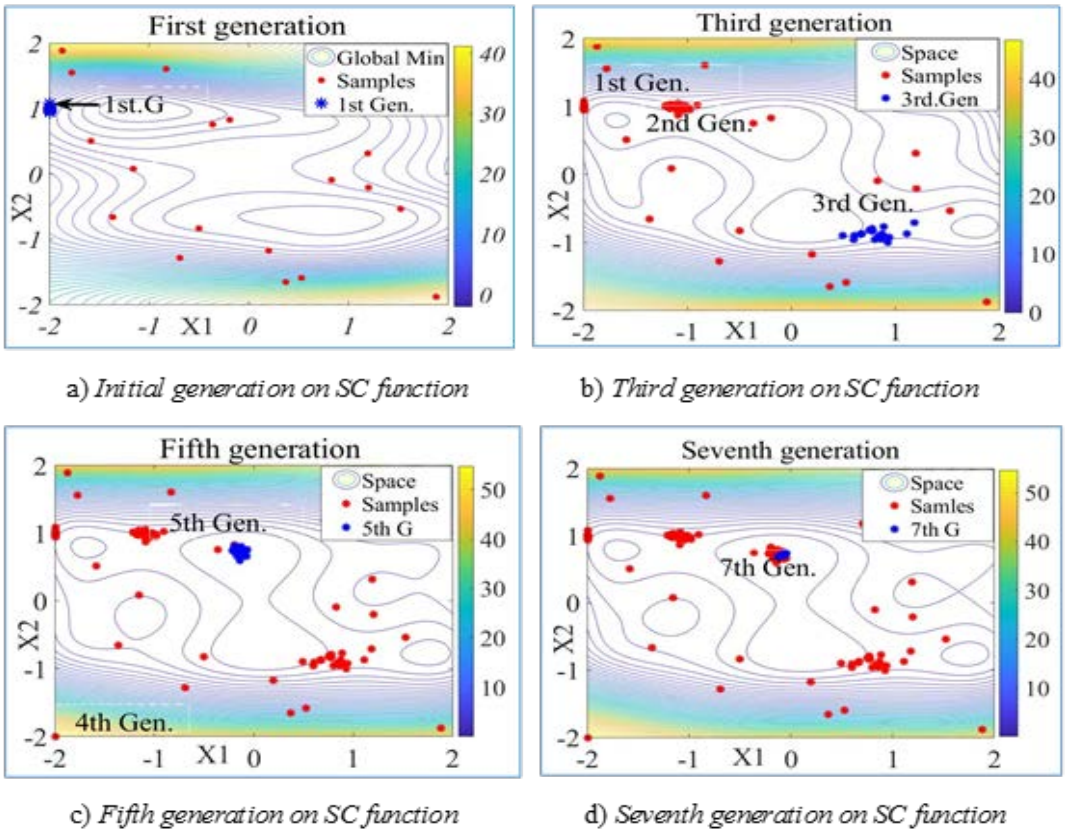


Figure 57. Flowchart of the Proposed K-BA Algorithm

6.5 The Optimization Process Based Kriging SM

The population-based Bat Algorithm (BA) has been successfully used in solving many global optimization problems due its ease of implementation and excellent ability of finding the global optimum solution, while the Kriging surrogate model (SM) has been effectively applied in many fields, particularly function approximation [216]. In this work, the Kriging SM and BA are combined to form a new algorithm, K–BA, for solving computationally expensive GO problems. Traditionally, BA requires a large number of function evaluations before converging to the global optimum, thereby impeding its application in solving computationally expensive problems. The new K–BA algorithm starts with an initial expensive set of sample points generated by a LHS to construct a Kriging model. Based on the Kriging model, BA is employed to search for better solutions on the SM to find promising solutions. Meanwhile, these candidates are evaluated and stored into the database, and repeated points are deleted.



a) *Initial generation on SC function* b) *Third generation on SC function*

c) *Fifth generation on SC function* d) *Seventh generation on SC function*

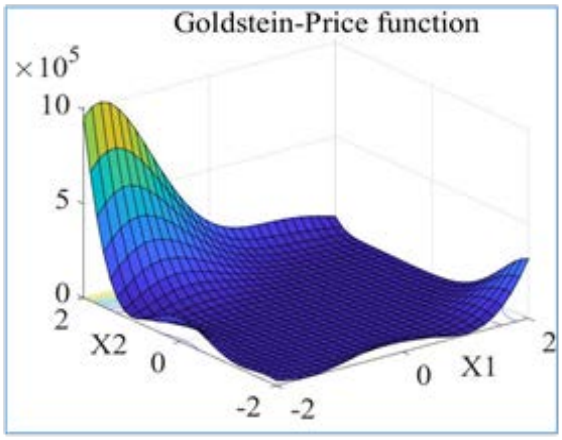
Figure 58.Generation and updating sample points on SC Function

During each iteration, all the sample points are used to refine the SM to get a better approximation. Since Kriging predictor can be used to acquire predictive promising regions and the unknown search space can be explored by the estimated MSE of Kriging [216], the Kriging SM is used in the optimization process for improved search efficiency. To demonstrate the K–BA search mechanism, the generation and update of sample points on SC function are illustrated in Figure 58 (a-d). The search strategy will ensure that better training points can be found based on the evolution of the surrogate model; then the Kriging SM is effectively re-built, and the unknown area of the design space can be efficiently explored. The estimated MSE of the Kriging SM is employed as an essential tool to examine the unexplored region of the design space. Furthermore, updating the database with extra sampling points will ensure that the promising region where the global optimum is located will not be missed. Figure 58 shows a progressive search generation after each of the four iterations to refresh the database and obtain more potential promising sampling points. At the beginning of the search, the region is too large to locate the global optimum; however, as the iteration continues, and the number of expensive sample points increases, the search space quickly shrinks to a local region around the global optimum. The Kriging model directed the optimization process to focus on a smaller area when the number of expensive points increased toward the global region. The surrogate model has assisted the BA to search only in the promising regions that accelerates convergence speed, thereby reducing computation time of the bat algorithm. In this search on the SC function, only 150 expensive sample points were used to find the global optimum. Exploration of the global optimal solution using the Kriging SM has increased search efficiency and reduced the number of expensive function evaluation.

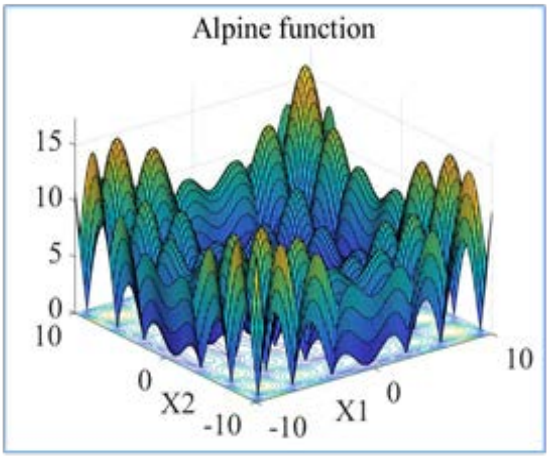
6.6 Standard Benchmark Functions

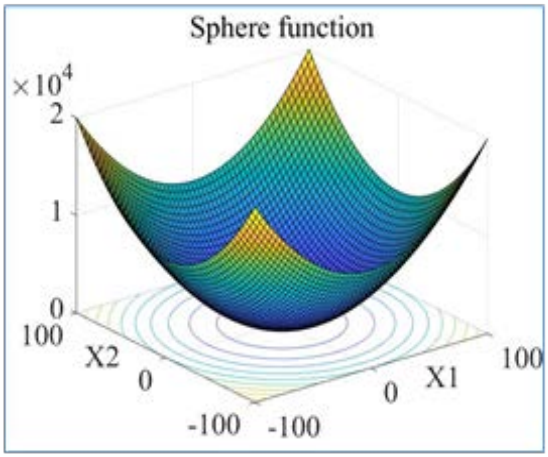
The convergence capability, search efficiency and robustness of any new global optimization method need to be verified using benchmark and representative optimization problems. The benchmarks representing different types of optimization problems, with continuous, discontinuous, unimodal, multimodal, low dimensional, and high dimensional objective functions; with continuous, integer and mixed variables; and with different types of constraints, are shown in Table 13 and Figure 59 (a-c). These include the widely used

optimization benchmark problems of the Banana's, Peak's, Levy's, Powell's, Aline's, Leon's, Sphere's and F16 functions [104, 217]. In addition, the new K–BA algorithm was also tested using five other constrained engineering optimization problems.



a) *Difficult to converge Unimodal function*





b) *Multimodal function*

c) *Unimodal function*

Figure 59. Samples of tested benchmark function

Table 13: Widely Used Optimization Benchmark Test Problems

N	Fun.	D	Analytic *f**	Search Space	Function Properties
f1	*Banana*	*2*	*0.0000*	*[-2 2]2*	*Differentiable , U-Modal*
f2	*GP*	*2*	*3.0000*	*[-2 2]2*	*Differentiable , M-Modal*
f3	*SC*	*2*	*−1.0320*	*[-2 2]2*	*Differentiable , M-Modal*
f4	*Shubert*	*2*	*−186.7309*	*[-10 10]2*	*Differentiable , M-Modal*
f5	*Peaks*	*2*	*− 6. 5511*	*[-4 4]2*	*Differentiable , M-Modal*
f6	*Shekel*	*4*	*−10.1532*	*[0 10]4*	*Differentiable , M-Modal*
f7	*Dixon - price*	*4*	*0.0000*	*[-10 10]4*	*Differentiable , U-Modal*
f8	*Powell*	*6*	*0.0000*	*[-4 5]6*	*Differentiable , U-Modal*
f9	*HM6*	*6*	*−3.3220*	*[0 1]6*	*Differentiable , M-Modal*
f10	*Leon*	*8*	*0.0000*	*[-1.2 1.2]8*	*Differentiable , U-Modal*
f11	*Alpine*	*10*	*0.0000*	*[-10 10]10*	*Differentiable , U-Modal*
f12	*Sphere*	*15*	*0.0000*	*[-5.12 5.12]15*	*Differentiable ,U-Modal*
f13	*HM16*	*16*	*25.8750*	*[-1 1]16*	*Differentiable , U-Modal*
f14	*G 11*	*2*	*0.7500*	*[-1 1]2*	*Constrained*
f15	*G15*	*3*	*961.7150*	*[0 10]3*	*Constrained*
f16	*TSD*	*3*	*0.0126665*	-	*Constrained*
f17	*WED*	*4*	*1.724852*	-	*Constrained*
f18	*SRD*	*7*	*996.34816*	-	*Constrained*

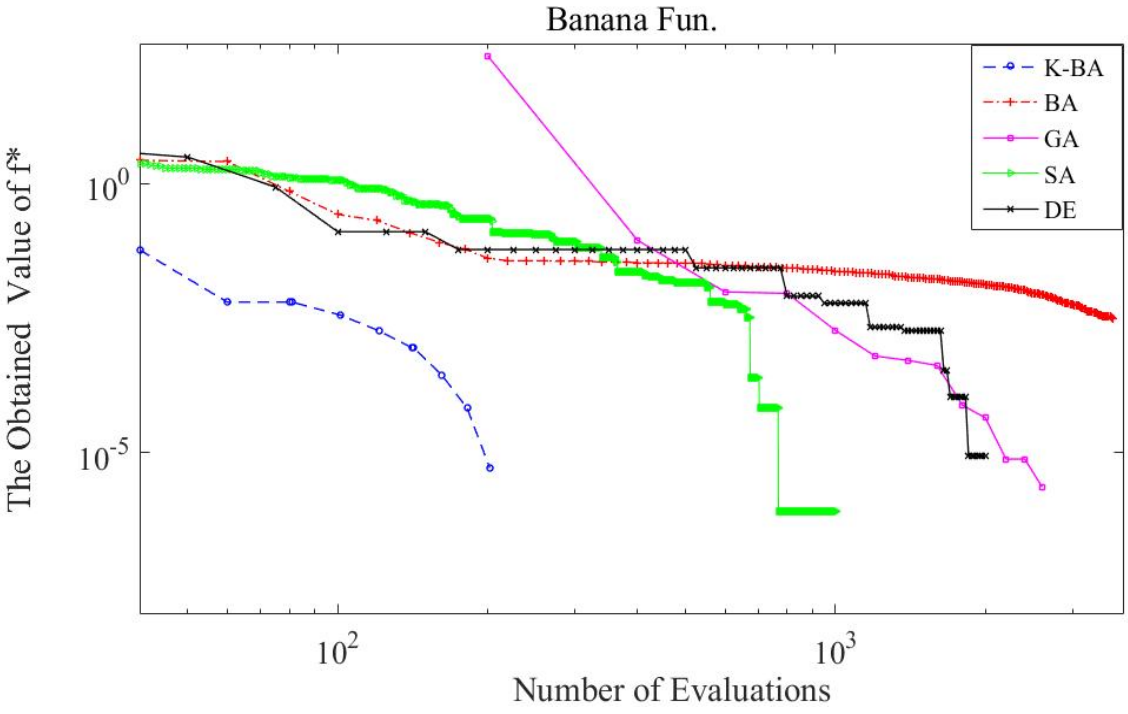


Figure 60.Test Function # 1

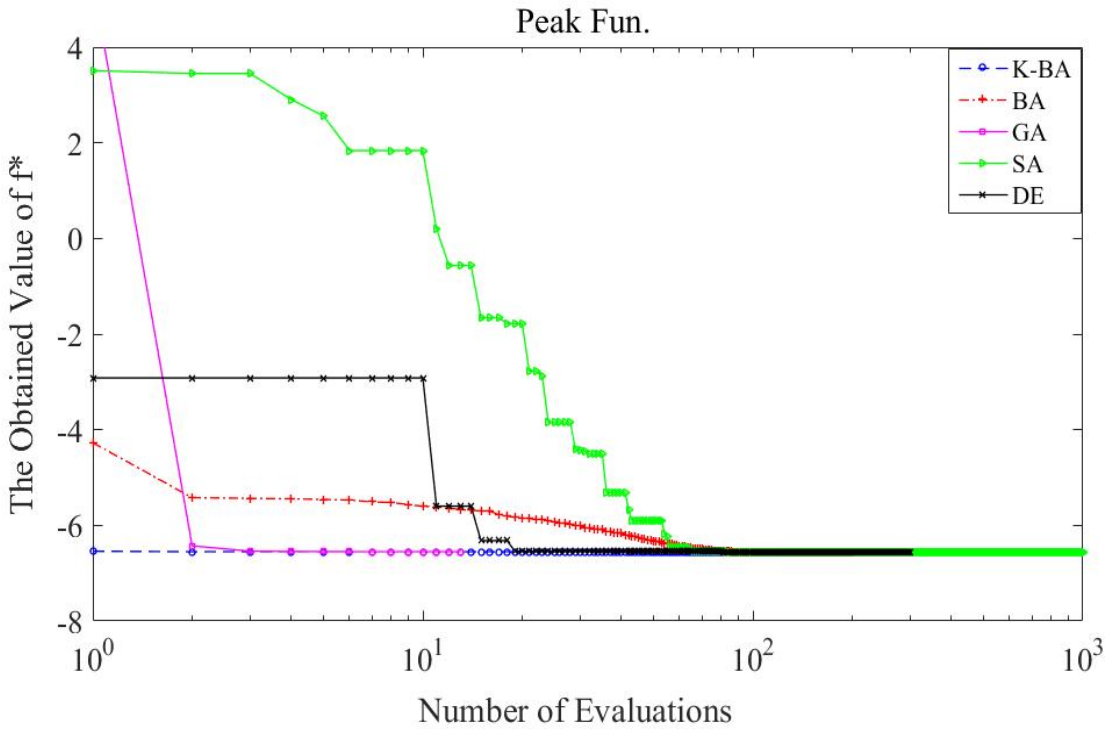


Figure 61.Test Function # 2

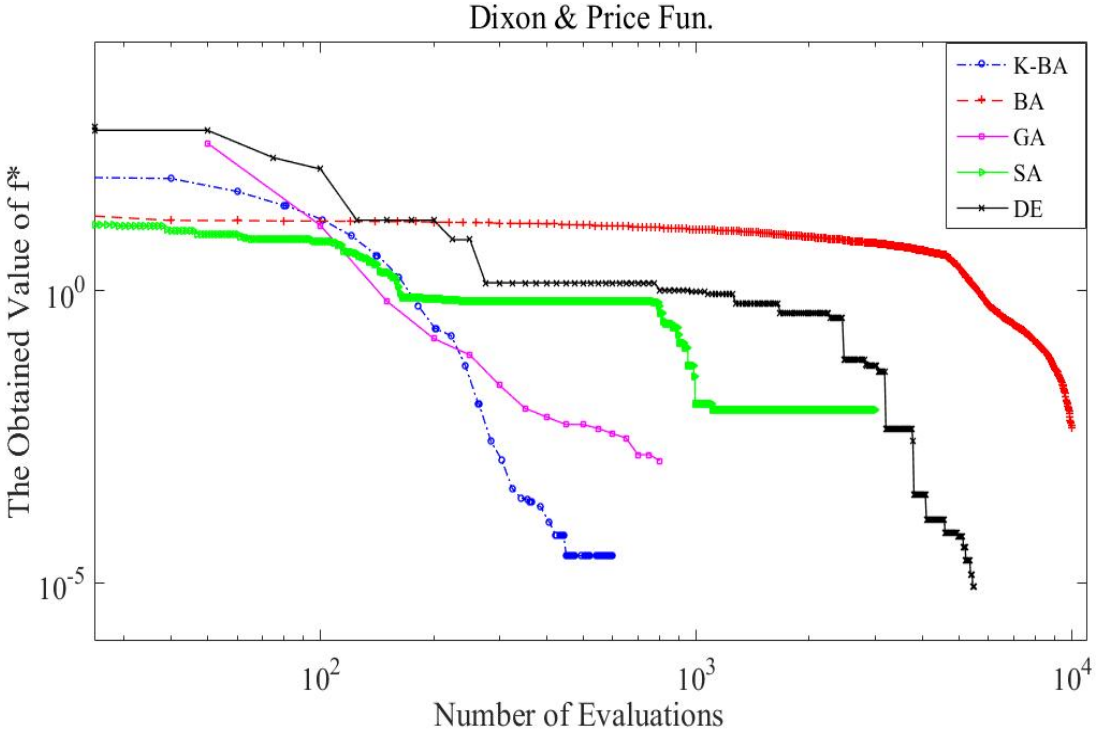


Figure 62. Test Function # 3

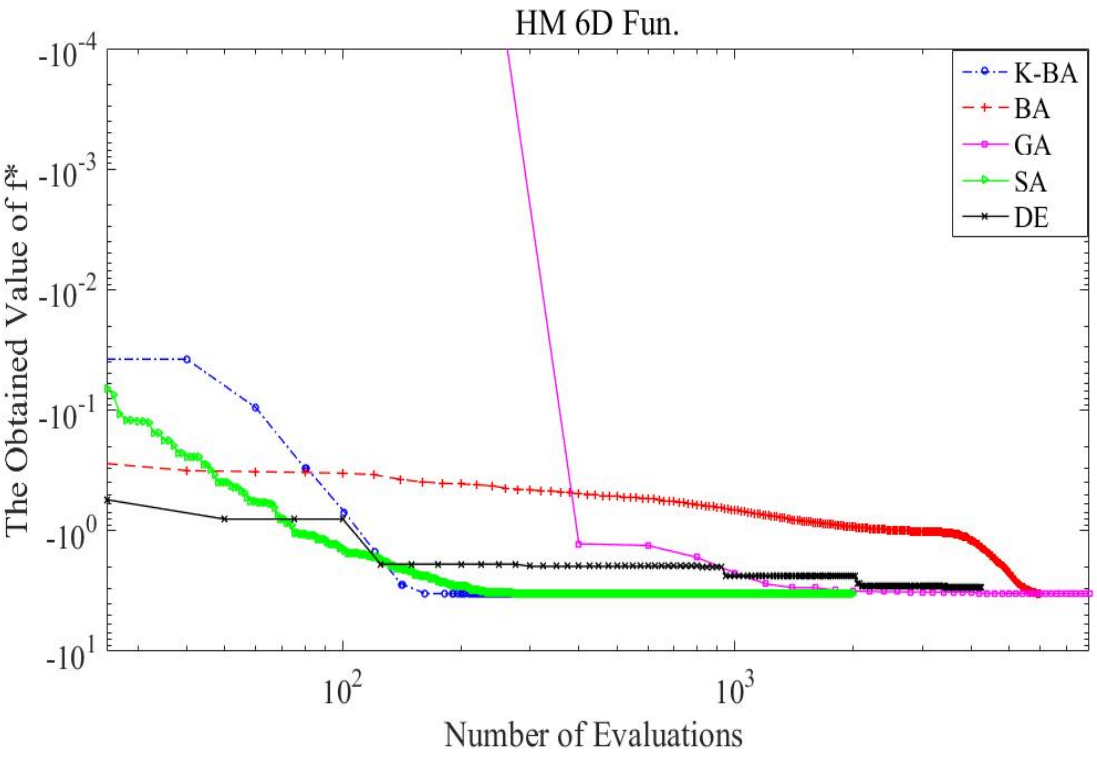


Figure 63.Test Function # 4

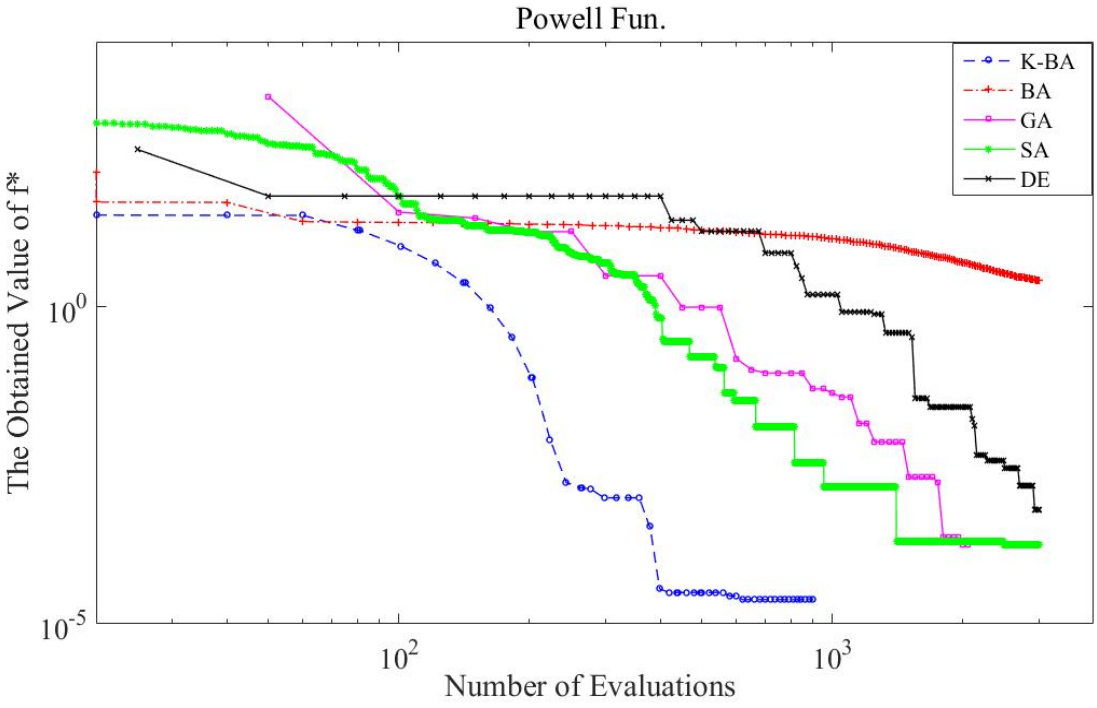


Figure 64.Test Function # 5

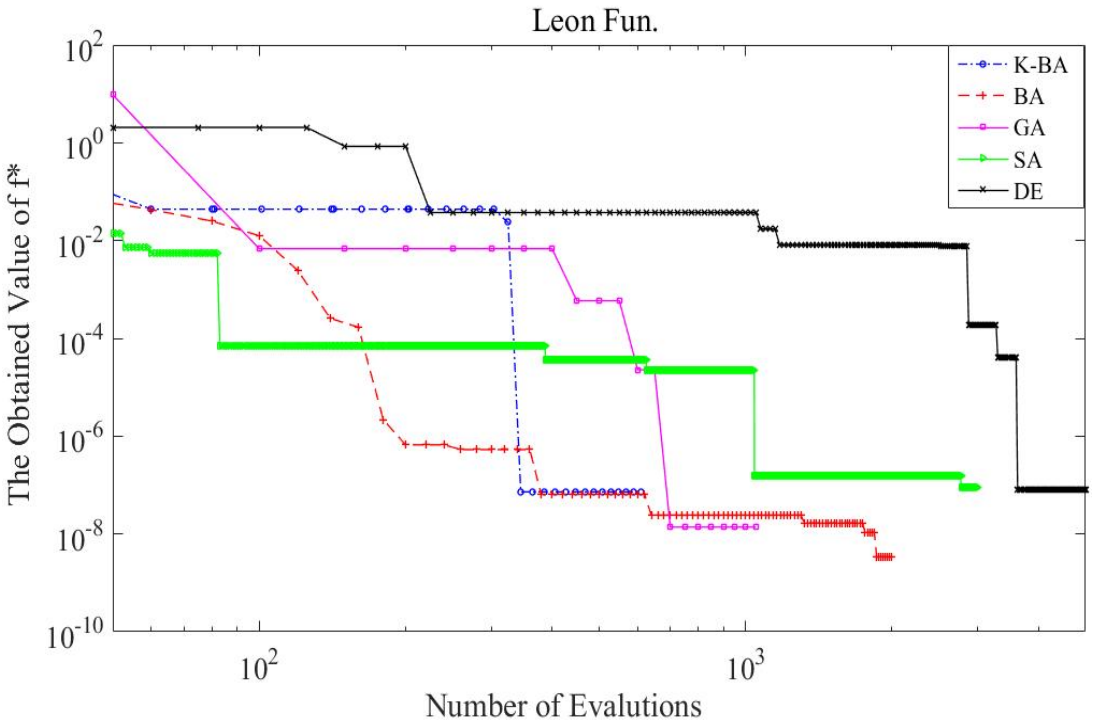


Figure 65.Test Function # 6

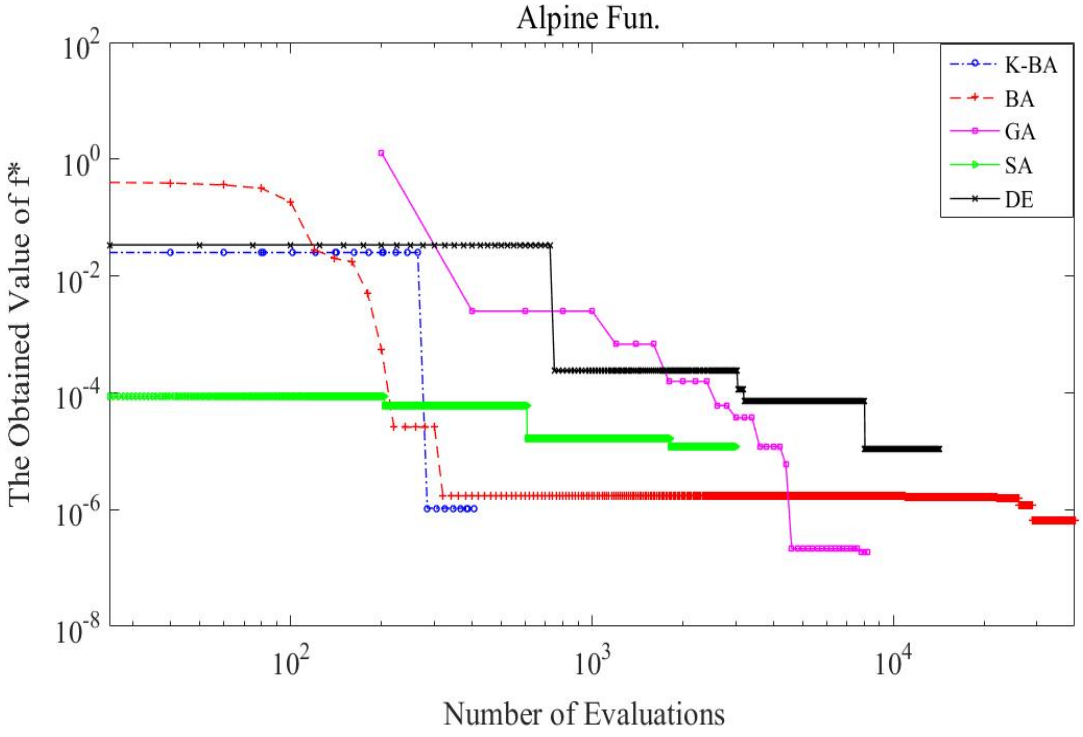


Figure 66. Test Function # 7

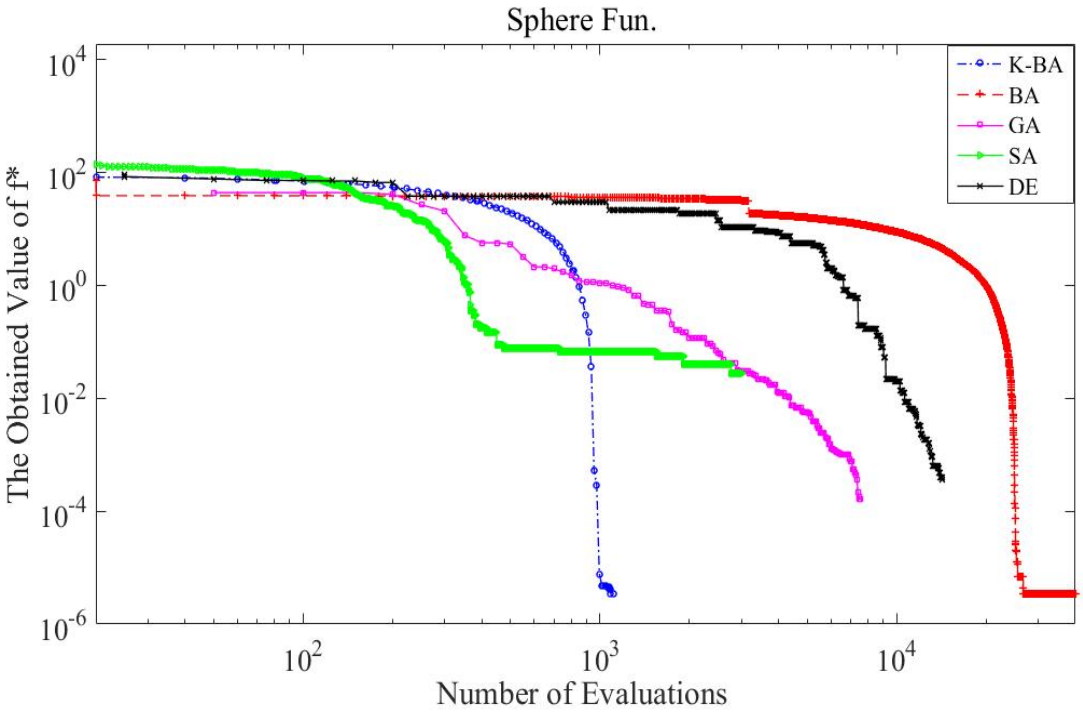


Figure 67. Test Function # 8

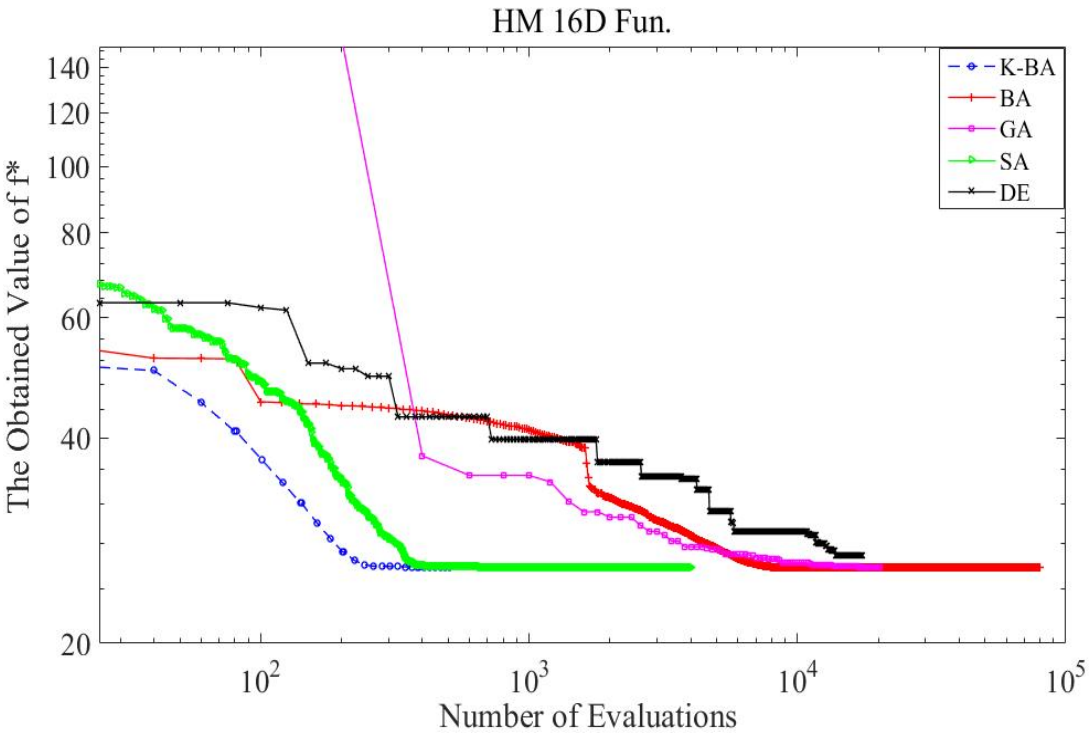


Figure 68.Test Function # 9

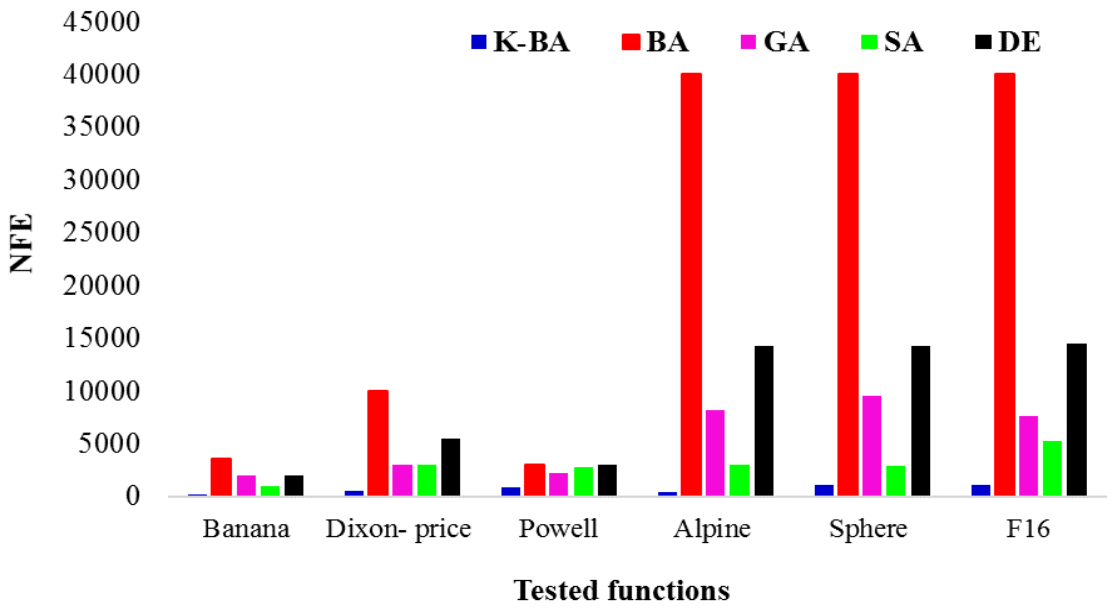


Figure 69.NFE of K-BA Vs (BA, GA, SA and DE) for Tested Functions

6.7 Experimental Results and Discussions

The main motivation for introducing the new K–BA was to improve search efficiency of BA in solving CEBB. The algorithm has been coded using MATLAB R2015b and tested on a PC with a 3.4GHz Core i7 processor, 16 GB RAM, and Windows 8. The optimization results were obtained from multiple runs, including the best, median and worst minimum values, and needed NFE in average, as shown in Table 2. The used stopping criteria were either up to 200 (for 2D problems), 500 (for 6D problems), and 1000 NFEs (for over 10D problems), respectively, or a successive value difference of 10−6. To illustrate the advantage of the new K–BA algorithm, the performance of this algorithm and four benchmark GO algorithms, GA, SA, Differential Evolution (DE) and the original BA, are compared using benchmark optimization problems with 2 to 16 design variables, as listed in Table 13. Table 14 listed the statistical search results of all five algorithms under comparison. The proposed K–BA showed superior performance on most test problems, with significantly reduced NFE on the computation intensive objective function. The different complexity of these test problems led to different degree of enhancements, but improvement on the rate of convergence of the new K–BA method showed superior search efficiency. Figure 60 to Figure 68 showed the convergence trend of different test problems

with 2, 6, 10 and 15 variables. The NFE required by the K–BA is significantly lower than the other four as shown in Figure 7. For instance, given the convergence accuracy of 10^{-5} and higher, the new K–BA method required only 202 expensive sample points, counted as NFEs, while BA, GA, SA and DE needed 3,600, 2,050, 998 and 2,000 NFEs, respectively for Function #1. Similarly, for Function #12, K–BA needed 1,108 NFEs to reach the global optimum, while BA, GA, SA and DE required 40,000, 9,500, 2,897 and 14,250 NFEs, respectively. These results are illustrated in Figure 69.

With significantly reduced NFEs required on the computation intensive objective function, the new K–BA method is able to serve the need for solving CEBB GO problems. The new K–BA method performed equally well for GO problems of different sizes (or numbers of design variables). The computation accuracy, search efficiency and robustness of all compared methods on the chosen functions are given in Table 14. These test results showed superior performance of K–BA on computationally intensive problems with demanding objective function evaluation.

Table 14: Results on Unconstrained Optimization Problems (Obtained f*)

	K-BA		BA		GA		SA		DE	
	Min	**NFE**	**Min**	**NFE**	**Min**	**NFE**	**Min**	**NFE**	**Min**	**NFE**
Banana	5.250E-6	202	0.0031	3600	3.20E-7	2250	1.37E-6	900	7.22E-6	2000
GP	3.0001	201	3	3600	3	2550	3.0001	1767	3.0001	2000
SC	-1.0316	150	-1.0316	1000	-1.0320	1550	-1.0320	1093	-1.0318	1000
Shubert	-186.7309	150	-186.7309	1000	-186.7309	1150	-186.7309	1500	-186.7309	1000
Peaks	-6.551	203	-6.551	800	-6.551	800	-6.551	683	-6.551	750
Shekel	-10.1532	302	-10.1532	3000	-10.1532	2250	-10.1532	1316	-10.1530	3250
Dixon	2.88E-6	600	4.60E-6	10000	1.22E-6	3050	9.08E-4	2996	8.54E-6	5500
Powell	2.40E-5	902	2.56	3000	1.80E-4	2050	1.79E-4	2800	6.23E-4	3000
HM6	-3.3223	300	-3.313	6000	-3.2032	8200	-3.319	2001	-3.271	4250
Leon	7.26E-8	609	3.36E-9	2000	1.38E-8	4200	8.98E-8	2784	8.06E-8	5000
Alpine	4.61E-6	406	3.44E-6	40000	1.911E-7	8200	1.185E-5	3000	1.94E-5	14250
Sphere	2.21E-6	1108	1.22E-6	40000	4.00E-4	9500	0.0040163	2807	2.77E-4	14250
HM16	25.8752	1109	25.875	40000	25.8801	7600	25.8827	5250	26.0638	14500

Table 15: Results on Constrained Optimization Problems

Problem			**Algorithm**				
	D		**K-BA**	**BA**	**GA**	**SA**	**DE**
G 11	2	Obtained f*	0.768	0.9296	0.7624	0.7624	0.7546
		NFE	504	10000	3550	3784	4250
G15	3	Obtained f*	961.7987	962.241	961.897	961.1373	961.9831
		NFE	609	6000	3550	3943	4250
TSD	3	Obtained f*	0.012673	0.012680	0.097426	0.014118	0.013611
		NFE	513	10000	3550	2896	2500
WBD	4	Obtained f*	1.7373	1.7531	1.7341	1.7356	1.72790
		NFE	667	10000	3550	3567	4250
SRD	7	Obtained f*	2992.0476	2997.8144	2992.0476	2988.0051	3061.2478
		NFE	506	10000	14200	3878	4250

6.8 Further Tests Using Constrained Optimization Problems

The newly introduced K–BA algorithm has also been tested using a number of nonlinear constrained engineering optimization problems. These include the G11 and G15 functions that came from the constrained optimization problems used in [218]; and three other structural design applications, including the Tension/Compression Spring Design (TSD),Welded Beam Design (WBD), Pressure Vessel Design (PVD), and Speed Reducer Design (SRD) [73, 190]. These five constrained test problems have their objectives and constraints in computationally expensive black-box form. The test results are presented in Table 15 , and the iterative search processes are illustrated in Figure 8. The proposed K–BA method identified the global optimum within 667 evaluations of the original objective and constraint functions on all of the five cases whereas BA, GA SA, and DE needed higher NFEs as shown in Table 15 and Figure 72. According to results from previously published performance tests lower [188, 190, 219], the obtained optima are sufficiently accurate, and the recorded NFEs are much lower. Furthermore, the K–BA search has been repeated ten times and subsequently applied on three of these cases to assess its robustness with the results shown in Table 4. In most cases, the K–BA method is able to obtain an accurate global solution with relatively low NFEs, showing outstanding computation efficiency and robustness.

Table 16: Summary of results obtained by K-BA on G15, TSD and SRD

Run.	**G15**		**TSD**		**SRD**	
	NFE	**Obtained *f****	**NFE**	**Obtained *f****	**NFE**	**Obtained *f****
No.1	*575*	*961.7441*	*502*	*0.0126664*	*503*	*2994.8493*
No.2	*569*	*962.9565*	*497*	*0.0127451*	*459*	*3001.4681*
No.3	*443*	*963.1568*	*511*	*0.0126717*	*478*	*2995.5570*
No.4	*521*	*961.7156*	*498*	*0.0126655*	*513*	*2995.4695*
No.5	*601*	*961.7792*	*491*	*0.0127865*	*496*	*2997.4988*
No.6	*597*	*961.8442*	*478*	*0.0126655*	*489*	*2998.5968*
No.7	*604*	*961.7977*	*517*	*0.0126670*	*501*	*2994.6535*
No.8	*607*	*962.0478*	*486*	*0.0126698*	*513*	*2997.0597*
No.9	*638*	*961.7161*	*507*	*0.0131654*	*504*	*2997.0869*
No.10	*711*	*961.7245*	*513*	*0.0126736*	*499*	*2999.5458*

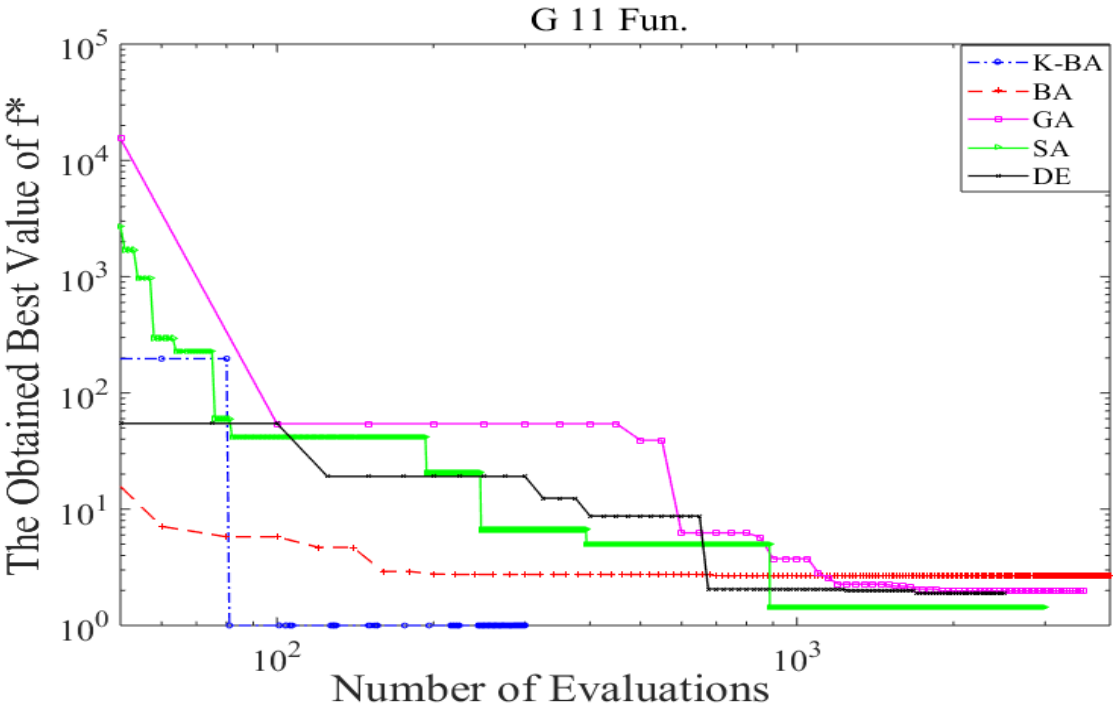


Figure 70.Convergence History of G11

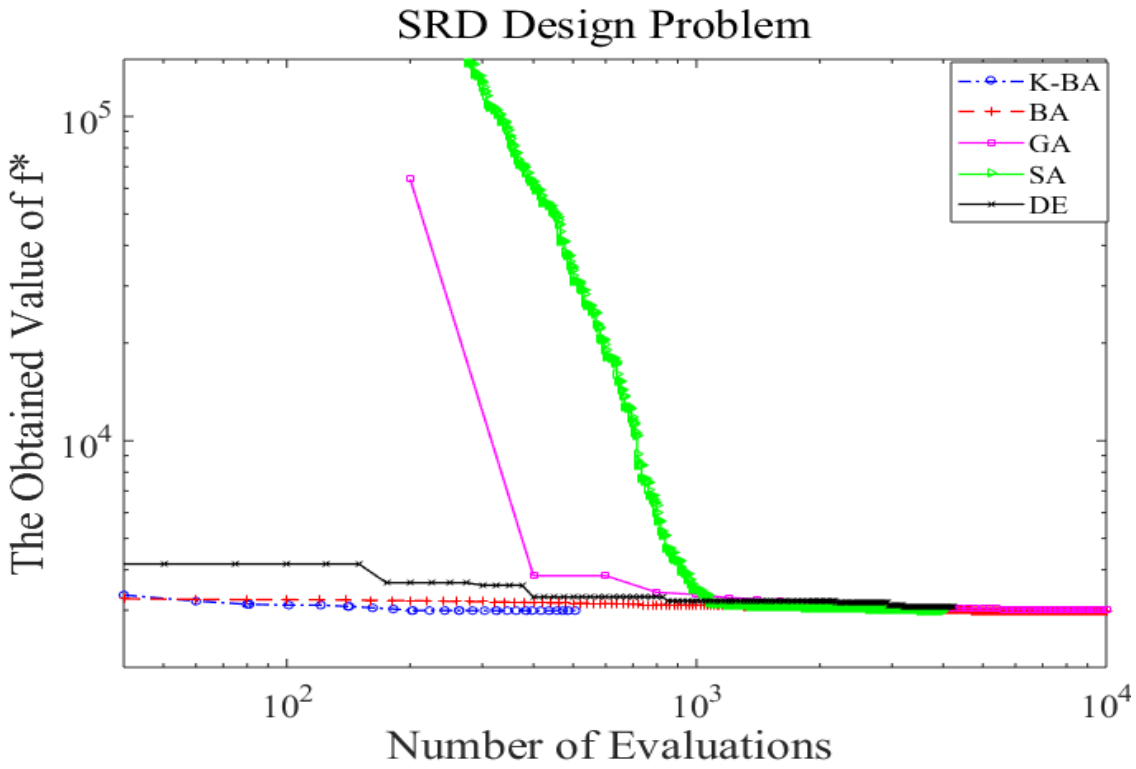


Figure 71.Convergence History of SRD

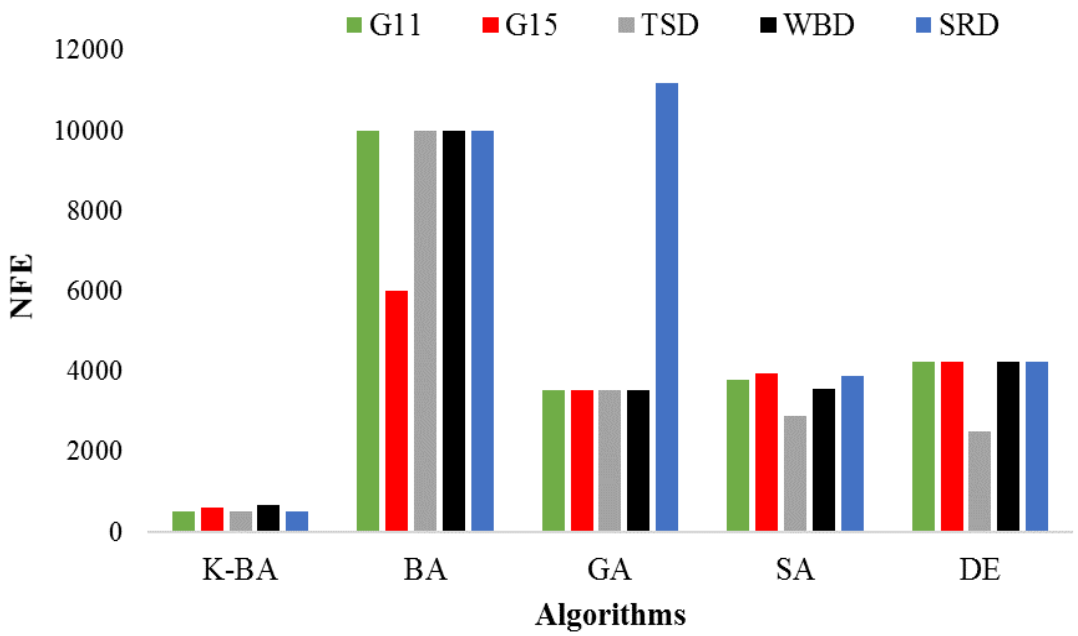


Figure 72. Number of function evaluations required by each method

6.9 Design Optimization of Floating Offshore Wind Turbine Platform

FOWT is an effective mean to utilize the vast space and wind resource in deep waters for generating electric power. However, the floating platform needs to be properly designed to maintain certain level of stability in the water, and to prolong the operational life of the wind turbine [220]. This section applies the new K-BA GO method to the FOWT design optimization application, and also uses this application to test the search capability and performance of various GO search techniques following the design method used in [194]. In the present study, a ballast-stabilized platform with a deep draft, also called spar buoy platform, as shown in Figure 73 is selected.

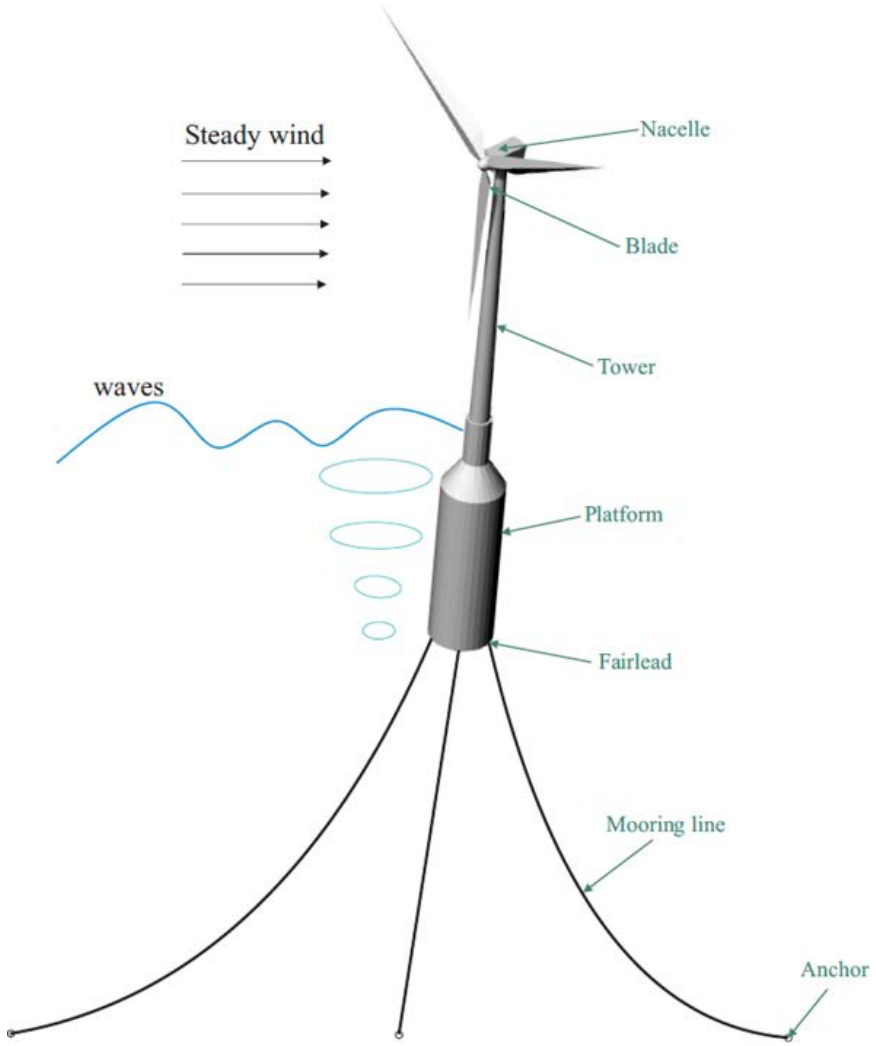


Figure 73.FOWT with a spar buoy platform

Proper selection of key platform design parameters, however, is a nontrivial task that involves several complex multiphysics models and interrelated design variables. In this work, identification of the optimal key platform parameters is done through numerical optimization. With the need of global optimization on the non-unimodal objective function and the CEBB nature of the problem, this platform optimization problem serves as an ideal test case on the effectiveness and relative performance of the new K-BA search method. The optimization is aimed at increasing the performance of a floating wind turbine by minimizing the nacelle acceleration using platform pitch and surge motions. These motions create extra loads on the wind turbine blades, causes fatigue in the drivetrain, and decreases the lifetime of the system [194]. In this study, a water depth of 300 *m* is applied universally.

Three sets of steady wind speeds, 8, 12, and 18 *m/s* are considered for calculating the aerodynamic loading on the NREL 5 MW offshore wind turbine [221]. Moreover, an irregular sea state is included based on a JONSWAP wave spectrum with significant wave heights of 2.5, 3.4, and 4.9 *m*, corresponding to the three wind speed conditions [194]. For the irregular wave spectrum, the frequency resolution of 0.027 rad/sec over the range of $0.25 \leq \omega \leq 2$ rad/s is defined. The objective function of this optimization problem is the average of the performance objective function calculated for the three environmental conditions. This optimization problem is both computationally expensive and in black box form.

To evaluate the stability of the floating system in the given environmental condition, the linearized aerodynamic characteristics of the 5 MW wind turbine, and the linearized hydrodynamic characteristics of the platform and mooring line must be collected in a frequency domain dynamic model. The software tool FAST [193] is used to calculate the linearized wind turbine mass, stiffness, and damping matrices in a given wind speed. In addition, the platform added mass, damping and hydrostatic as well as mooring line stiffness are calculated using another design tool WAMIT [222] and quasi-static mooring subroutine of the FAST, respectively. The linearity of the simplified dynamic system is exploited to define the response amplitude operator (RAO) of nacelle acceleration (Eq. 6.8) at each sea state condition based on the complex responses of the linearized frequency domain equation of motion (Eqs. 6.6 and 6.7).

$$-\omega^2 M_{total}(\omega)\hat{Z}(\omega) + i\omega B_{total}(\omega,\zeta)\hat{Z}(\omega) + C_{total}\hat{Z}(\omega) = \hat{X}(\omega) \tag{6.6}$$

$$[RAO_1(\omega) :: RAO_6(\omega)] = [-\omega^2 M_{total}(\omega) + i\omega B_{total}(\omega,\zeta) + C_{total}]^{-1}\hat{X}(\omega) \tag{6.7}$$

where, M_{total}, B_{total}, and C_{total} are the total mass matrix, damping matrix, and stiffness matrix of the FOWT respectively. $\hat{Z}(\omega)$ is the complex amplitude vector for the platform displacement and $\hat{X}(\omega)$ is the first-order wave excitation vector calculated by WAMIT. The complex form of nacelle acceleration *RAO* is given by Eq. 6.8. The performance metric for this study is defined as the standard deviation of nacelle acceleration (Eq. 6.9).

$$\mathrm{RAO}_{a_{nac}}(\omega) = -\omega^2(\mathrm{RAO}_1(\omega) + \mathrm{RAO}_5(\omega) z_{nac}) \tag{6.8}$$

$$\sigma_{a_{nac}}(\omega) = \sqrt{\int_0^{\infty} |\mathrm{RAO}_{anac}(\omega)|^2 S(\omega)\, \mathrm{d}\omega} \tag{6.9}$$

where z_{nac} is the wind turbine hub height (90 m) and $S(\omega)$ is the spectral density of the waves at prescribed sea states. Note that $\mathrm{RAO}_{a_{nac}}(\omega)$ is the fore-aft response amplitude operator of nacelle acceleration and $\sigma_{a_{nac}}(\omega)$ is the standard deviation of this response.

The design optimization seeks the best performance of the wind turbine platform measured by the performance metric defined in Eq. 6.9 with respect to a number of key design variables: cylinder draft, H_I, cylinder radius, R_I, top tapper ratio, T_I, and a catenary mooring system, X_M, as given in Table 17 and Figure 74. For the spar buoy platform, the free board (FB) of 5 m is considered. The mooring system design variable, X_M, defines three-angled taut line and slack line configurations for the spar buoy platform. The mass of support structure is determined using a wall thickness of 50 mm with a steel density of 8050 kg/m^3. More details about the platform parametrization, mooring and anchor system, and the linearized dynamic model can be found in [194].

Table 17: Geometric Design Variables of the Platform

Variable	Description	Min.	Max.
HI	*Cylinder draft*	*2 m*	*150 m*
RI	*Cylinder radius*	*3 m*	*25 m*
TI	*Top taper ratio*	*0.2*	*2*
XM	*Mooring system*	*0*	*2*

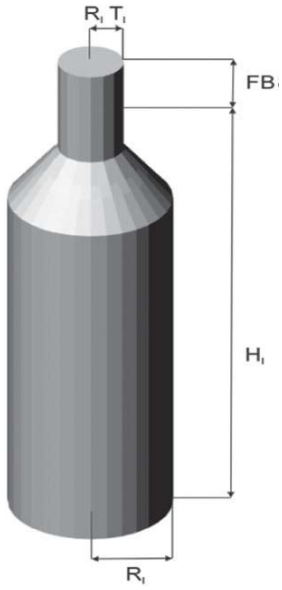


Figure 74. Design characteristics of the spar buoy platform.

Here, R_I is the cylinder radius, H_I is cylinder draft, T_I is the top tapper ratio, and F_B is the platform free board (5 *M*).

In the current study, a number of design constraints are also applied to the objective function. To make sure about the stability of the platform and wind turbine, the standard deviation of the nacelle acceleration (performance metric) is limited to 1 m/s^2. Moreover, to avoid over-turning of the platform, the steady-state pitch angle of the platform should be less than 10° [220]. The expression of this constraint is given in the following equation.

$$\zeta_5 = \frac{F_{hrust.z_{nac}} + M_{Mooring_5} - M_{ballast}}{\rho g \forall z_{CB} - M_t g z_{CG} + \rho g\, Ixx - C_{mooring5,5} + C_{5,1} z_{\text{fair}}} \tag{6.10}$$

where, F_{thrust} is the steady thrust load of the wind turbine, $M_{mooring5}$ is the mooring line pitching moment at the maximum wind turbine thrust; $M_{ballast}$ is the pitching moment due to stabilizing ballast mass; $\forall$ is the platform displacement; z_{CB} is the center of buoyancy location; M_t is the total mass of the system; z_{CG} is the center of gravity location; I_{xx} is the platform water plane moment of inertia in pitch motion; $C_{mooring5,5}$ is the mooring lines stiffness in pitch motion; $C_{5,1}$ is the mooring line stiffness in pitch-surge motions; and z_{fair} is the fairlead depth in pitch motion. For more details about the design constraints, the reader is referred to the study presented in [194].

The design objective and constraints to the optimization are modeled in the complex and implicit forms. The objective function is in non-unimodal form with many local optima, requiring a GO algorithm to solve the function. The evaluation of the objective and constraint functions also requires intensive numerical computation, further requiring the GO search algorithm to be computationally efficient with less required NFEs. For comparison, four other benchmark GO algorithms, including BA, GA, SA, and DE, have also been used in the search of global design optimum solution. The test results from these five GO algorithms are shown in Table 18.

Table 18: Results of the Wind Turbine and Algorithm Performance

Algorithms									
K-BA		**BA**		**GA**		**SA**		**DE**	
f*	**NFE**	***f****	**NFE**	***f****	**NFE**	***f****	**NFE**	***f****	**NFE**
***0.0011*5**	*540*	***0.00129***	*600*	***0.00*127**	*600*	***0.00518***	*602*	***0.00127***	*600*
x_1^*	149.7	-	*147.6*	-	*148.11*	-	*148.14*	-	*149.2*
x_2^*	*23.98*	-	*23.95*	-	*23.58*	-	*24.11*	-	*23.68*
x_3^*	*0.30*	-	*0.19*	-	*0.38*	-	*0.31*	-	*0.29*
x_4^*	*1.97*	-	*1.92*	-	*1.89*	-	*1.37*	-	*1.71*

6.10 K-BA Performance on (FOWT) Platform Application

In this four variable optimal design application, the capability and the performance of the new K–BA method for solving real life computationally expensive black-box optimization problems is tested. Table 18 presents the average of 20 different run results from K–BA against the four well-known benchmark GO algorithms. The same optimization parameters as in the previous test section were used with population size set as 20. The statistical results presented in Table 18 showed that the new K–BA method produced the best solution with considerably fewer NFEs. The algorithm found the promising regionwith100NFEs and quickly converged to the global optimum after only 540 NFEs, while the other four algorithms largely stopped at less accurate results after 600 NFEs. The new K–BA also

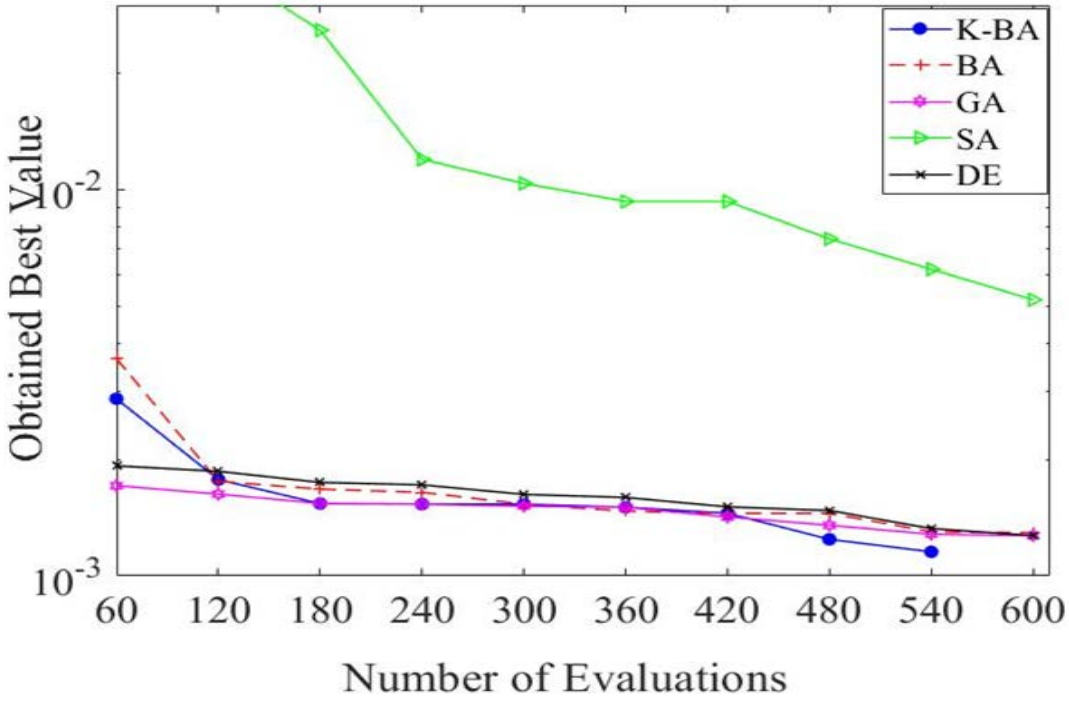


Figure 75. Convergence rate of wind turbine design

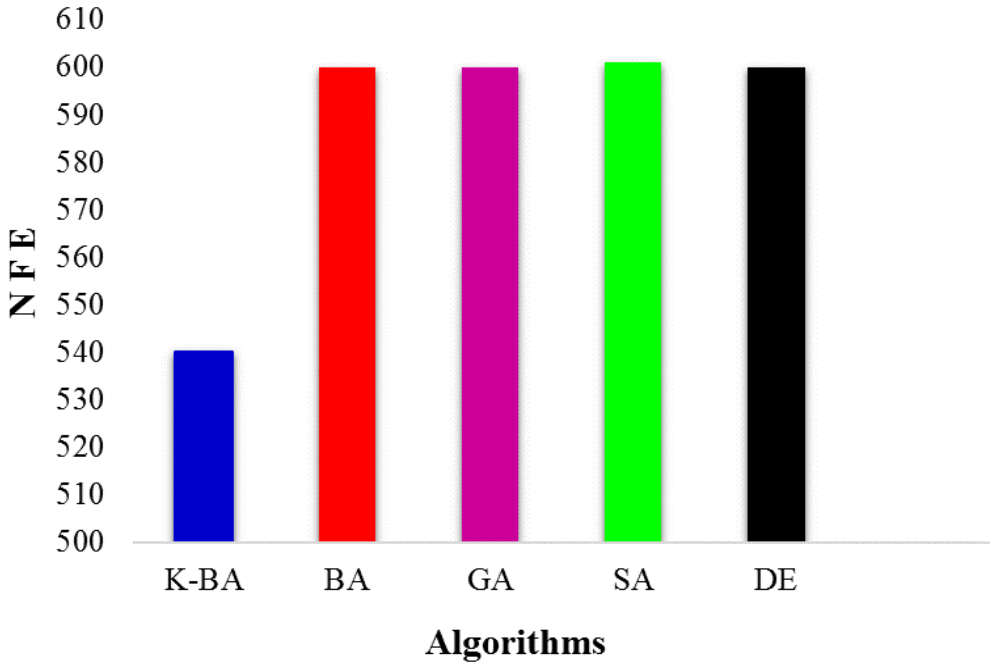


Figure 76. NFE required by each method

outperformed the original BA. Besides the best value of the obtained design optimum, another important factor is the convergence rate that is reflected by the NFEs or computation time. The results shown in Figure 75 and Figure 76 showed that the newly K–BA is able to locate the global optimum with considerably lower NFEs. The newly K–BA algorithm showed robustness in handling the computationally expensive, real-life engineering application problem with consistent and reliable optimization results.

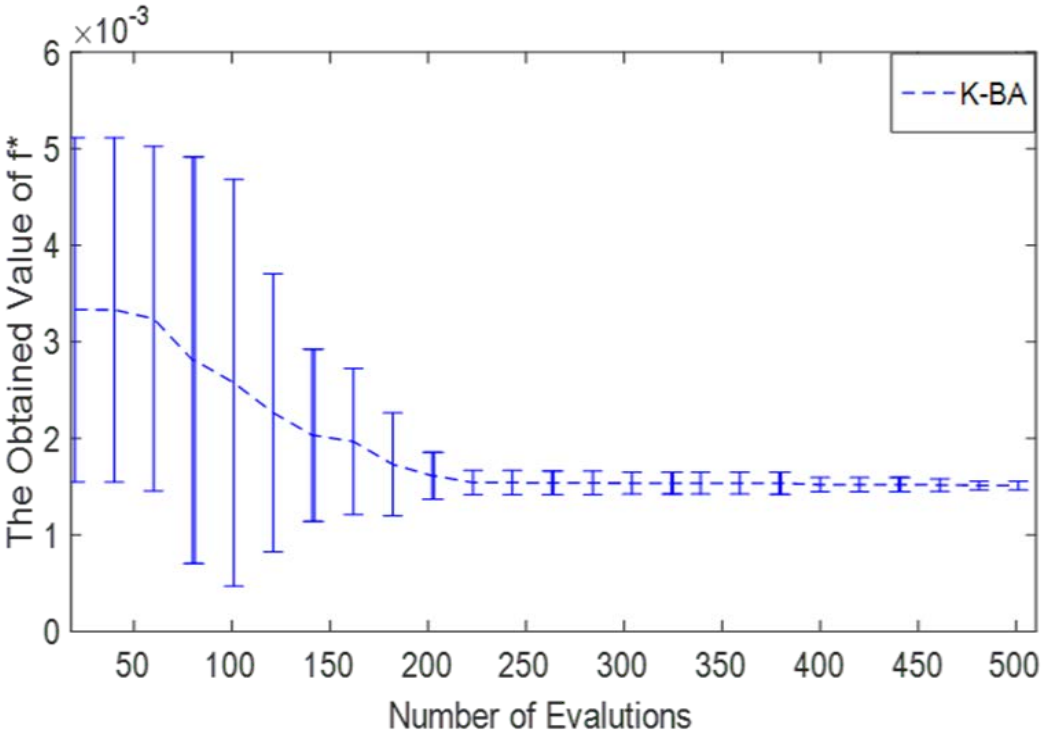


Figure 77. Standard deviations (error bars) of K-BA algorithm on wind turbine optimization problem

As a common practice in testing global optimization algorithms, all optimization results shown in this study are results of 20 different runs. This approach better assures the validity of the global optimization search. These results are normally distributed over a narrow band depending upon the robustness of the algorithm and the stage of the stochastic search. The average value of the results is adopted and the standard deviation is used as a measure on the convergence of the search. This is shown using the K-BA test example as shown in Figure 77. In this case, a comparison on the efficiency of the K-BA algorithm for the wind turbine optimization problem is shown using the varying means and the standard deviations

of the 20 different runs. These results indicated that the newly proposed search method is capable of solving the computationally expensive wind-turbine optimal design problem more efficiently than the standard BA using less computational effort.

6.11 Summary

In this work, a new global optimization search method, named K–BA, that combines Kriging surrogate modelling with the Bat Algorithm for solving CEBB global optimization problems has been introduced. The Kriging surrogate model is used to find the promising region where global optimum is mostly likely located, and to alleviate complexity and computational time by replacing expensive simulation and analysis on the original CEBB problem. The Kriging model also covers the entire design space and ensures that the true global optimum will not be readily missed. During the search, additional sample points are used to supplement the Kriging model, and the estimated mean square error of Kriging is used to guide the search using BA in the promising region. The new K–BA algorithm is compared with other well-known GO algorithms, including BA,GA,SA, and DE through experiments on a set of standard optimization benchmark problems with 2 to 16 design variables to examine their relative search capability, efficiency, and robustness. The statistical results on the set of tested functions showed that for each problem tested, the proposed K–BA achieved superior accuracy compared to the other tested GO methods. In addition, the proposed K–BA could reach the optimum with less iterations for every tested problem, leading to considerably (typically more than 50 percent fewer) objective function evolutions. A wind-turbine design optimization problem, as a real-life design optimization application, was also used as an additional test case. Overall, test results consistently proved the suitability and superior capability of the newly proposed K–BA in solving complex CEBB global optimization problems with significantly reduced number of original objective function evaluations.

Chapter 7. Conclusions and Future Work

7.1 Conclusions

Rapid advances in computer modeling and simulation tools and computing hardware have made complex simulation model-based design optimization more viable today. Continuous improvements on stochastic and natural-inspired global optimization (GO) techniques lead to more efficient and robust GO search methods, including Divided Rectangles (DIRECT), Differential Evolution (DE), and Bat Algorithm (BA). However, solving complex, computationally intensive, "black-box" global optimization problems demands new search algorithms that can quickly identify the global optimum with a much lower number of evaluations of the computationally expensive objective and constraint functions. Surrogate Model (SM), or metamodeling-based global optimization techniques have been introduced to address this issue. In these algorithms, various surrogate models, including Kriging, radial basis functions (RBF), multivariate adaptive regression splines (MARS), and polynomial regression (PR), are built using limited samplings on the original, "expensive" objective/constraint functions and are used for a significant amount of "cheap" objective/constraint function evaluations in the search of the global optimum to dramatically reduce needed computation time. The effective integration of surrogate modeling and stochastic/natural-inspired GO techniques, which can overcome the challenges in dealing with the complex, computationally intensive, "black-box" global optimization problems, form the objective of this research.

In this work, improvements have been made on three widely used global optimization algorithms, namely Divided Rectangles (DIRECT), Differential Evolution (DE), and Bat Algorithm (BA), by integrating appropriate surrogate modeling methods to increase the computation efficiency of these algorithms to support MBD. The superior performance of these new algorithms in comparison with their original counterparts are shown using commonly used, optimization algorithm testing benchmark problems. This leads to the new algorithms of:

- Kriging Surrogate Model Improved Divided Rectangles Algorithm (K-DIRECT)

- Radial Basis Function Surrogate Model Improved Differential Evaluation Algorithm (RBF-DE), and
- Kriging Surrogate Model Improved Bat Algorithm (K-BA)

The Addition of surrogate modeling has considerably improved the search efficiency of the DIRECT, DE, and BA algorithms with significant reduction on the Number of Function Evaluations (NFEs). The newly introduced algorithms are then applied to a complex engineering design optimization problem, the design optimization of floating wind turbine platform, to test its effectiveness in real-world applications. These newly improved algorithms were able to identify better design solutions using considerably lower NFEs on the computationally expensive performance simulation model of the design. The resulting algorithms proved to be effective for solving complex and computationally intensive black-box global optimization problems, and forms a foundation for future research in this area.

7.1.1 Kriging SM Improved Divided Rectangles Algorithm (K-DIRECT)

A new K-DIRECT method is proposed to get an accurate approximation of the optimized problem function with respect to the limited number of sample points. This work aims at increasing computational efficiency by modifying the original DIRECT Search GO algorithm using previously evaluated points by the DIRECT algorithm to build SM. Kriging SM is used to predict the original objective function in computationally expensive, black-box (CEBB) form. Following the Kriging SM construction, metamodeling evaluation is used to update the best function value for search using the DIRECT algorithm. This metamodeling guided DIRECT search strategy forms the new Kriging DIRECT algorithm (K-DIRECT). In this approach, the DIRECT algorithm only searches in the most promising region, and identifies the global optimum based on the best candidate solutions. In the thesis, the new algorithm and its implementation are explained. Tests on the robustness and efficiency of the newly proposed algorithm against the original DIRECT GO search method are carried out using a number of benchmark problems. The statistical results reported in Chapter Three showed the advantages of the new K-DIRECT algorithm.

The K-DIRECT algorithm obtained better results in most cases and consistently outperformed the original DIRECT search global optimization method in terms of NFE required, making it an ideal tool for identifying the optimal design using a complex, black-

box computer simulation objective function in a multidisciplinary design optimization. The algorithm performs well largely for low dimensional, unconstrained problems, leading to the introduction of more complex search algorithms found in the following sections.

7.1.2 RBF SM Improved Differential Evaluation Algorithm (RBF-DE)

Finding and designing optimization techniques to deal with computationally expensive problems require an effective and efficient methodology. This work proposes a series of modifications to the Differential Evolution (DE) algorithm. The proposed Radial Basis Functions SM in combination with Differential Evolutionary search (RBF-DE) is another GO algorithm based on metamodeling techniques. The RBF surrogate model is used to approximate the expensive objective function, while the DE algorithm employs a mechanism to identify the best combination of parameters in performing the search, such as differential rate, crossover probability, and population size. The proposed algorithm has been tested using benchmark functions and real-life practical applications and problems. The test results showed that the proposed algorithm outperform other optimization algorithms. The new algorithm is capable of converging to the global optimum in improved accuracy, number of function evaluations, and computation time.

7.1.3 Kriging SM Improved Bat Algorithm (K-BA)

A new Kriging-Bat Algorithm (K-BA) is introduced for solving CEBB global optimization problems with further improved search efficiency and robustness. A Kriging surrogate model is integrated with the BA to find the global optimum using a substantially reduced number of function evaluations of the original objective function. During the search, additional sample points are used to supplement the Kriging model, and the estimated mean square error of Kriging is used to guide the search using BA in the promising region. The new K-BA algorithm is compared with other well-known GO and MBGO algorithms, including GA, SA, DE and BA. Experiments on a set of standard optimization benchmark problems with 2 to 16 design variables were conducted to examine their relative search capability, efficiency, and robustness. A wind-turbine design optimization example was also used in the tests as a real-life design optimization application. Results of the statistical tests demonstrated the superior capability of the newly proposed K-BA in solving complex

CEBB global optimization problems with much reduced numbers of original objective function evaluations.

7.1.4 A Comparative Study on Nature-Based Global Optimization Methods in Complex Mechanical System Design

Advanced global optimization algorithms have been continuously introduced and improved to solve complex GO problems for which the objective and constraint functions can only be evaluated through computation intensive numerical analyses or simulations with a large number of design variables. The often implicit, multimodal, and ill-shaped objective and constraint functions in high-dimensional and "black-box" forms demand the search to be carried out using a low number of function evaluations with high search efficiency and good robustness. This work also investigated the performance of six recently introduced, nature-inspired global optimization methods: Artificial Bee Colony (ABC), Firefly Algorithm (FFA), Cuckoo Search (CS), Bat Algorithm (BA), Flower Pollination Algorithm (FPA) and Grey Wolf Optimizer (GWO) firstly to identify the most suitable GO algorithm to improve with the addition of SM, and secondly to introduce a real engineering design optimization problem as an example of a computationally-expensive black-box global optimization. These approaches are compared in terms of search efficiency and robustness in solving a set of representative benchmark problems in smooth-unimodal, non-smooth unimodal, smooth multimodal, and non-smooth multimodal function forms. In addition, four classic engineering optimization examples and a real-life complex mechanical system design optimization-problem floating offshore wind turbines design optimization are used as additional test cases representing computationally-expensive black-box global optimization problems. Results from this comparative study show that the ability of these global optimization methods to obtain a good solution diminishes as the dimension of the problem, or number of design variables increases. Although none of these methods is universally capable, the study finds that GWO and ABC are more efficient on average than the other four in obtaining high quality solutions efficiently and consistently, solving 86% and 80% of the tested benchmark problems respectively.

7.2 Future Work

Based on the experience gained and results obtained in this work, the following research may be considered for future work.

- **Improving the Kriging DIRECT Search Algorithm (K-DIRECT)**

K-DIRECT has the ability to find the global optimum of any low dimensional problem with the best function value of all tested problems. However, there is a limitation in using it for high dimensional problems. Hence, the K-DIRECT introduced in this thesis needs to be studied and extended for high-dimensional optimization problems and to verify its merits in reducing computation time generally.

- **Increasing the Efficiency of RBF-DE for High-Dimensional CEBB**

The surrogate-assisted sampling search should be improved, considering other surrogate models and integration mechanisms, and the search process needs to be optimized to further save CPU time and/or to reach greater performance accuracy.

- **Extension of the Kriging-Bat Algorithm (K-BA)**

Kriging-guided BA approach should be extended and modified for high dimension optimization problems. The proposed surrogate models used to assisted BA algorithm has the potential to be applied to the other swarm intelligence algorithms, such as ant/bee colony optimization (ACO/ABC) techniques as part of future studies due to their similarities.

- **Applying the New GO Methods to Different Applications**

Although surrogate-assisted GO algorithms are tested using several benchmark functions and a few engineering problems, it would be beneficial to test them over a wider scope of design applications. Various complex, black-box optimization problems are candidates for case studies that facilitate the identification of the areas that need further improvements.

References

1. Garg, H., *A hybrid PSO-GA algorithm for constrained optimization problems.* Applied Mathematics and Computation, 2016. **274**: p. 292-305.
2. Samora, I., et al., *Simulated annealing in optimization of energy production in a water supply network.* Water resources management, 2016. **30**(4): p. 1533-1547.
3. Garg, H. and S. Sharma, *Multi-objective reliability-redundancy allocation problem using particle swarm optimization.* Computers & Industrial Engineering, 2013. **64**(1): p. 247-255.
4. Jin, Y., *A comprehensive survey of fitness approximation in evolutionary computation.* Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2005. **9**(1): p. 3-12.
5. Jin, Y., M. Olhofer, and B. Sendhoff, *A framework for evolutionary optimization with approximate fitness functions.* IEEE Transactions on evolutionary computation, 2002. **6**(5): p. 481-494.
6. Buche, D., N.N. Schraudolph, and P. Koumoutsakos, *Accelerating evolutionary algorithms with Gaussian process fitness function models.* IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2005. **35**(2): p. 183-194.
7. Liu, B., Q. Zhang, and G.G. Gielen, *A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems.* IEEE Transactions on Evolutionary Computation, 2014. **18**(2): p. 180-192.
8. Ratle, A., *Kriging as a surrogate fitness landscape in evolutionary optimization.* AI EDAM, 2001. **15**(1): p. 37-49.
9. Ong, Y.S., P.B. Nair, and A.J. Keane, *Evolutionary optimization of computationally expensive problems via surrogate modeling.* AIAA journal, 2003. **41**(4): p. 687-696.
10. Sun, C., et al., *A two-layer surrogate-assisted particle swarm optimization algorithm.* Soft computing, 2015. **19**(6): p. 1461-1475.
11. Jones, D.R., *Direct global optimization algorithmdirect global optimization algorithm*, in *Encyclopedia of optimization*. 2001, Springer. p. 431-440.
12. Clausen, J. and M. Perregaard, *On the best search strategy in parallel branch-and-bound: Best-First Search versus Lazy Depth-First Search.* Annals of Operations Research, 1999. **90**: p. 1-17.
13. Abascal, F. and A. Valencia, *Clustering of proximal sequence space for the identification of protein families.* Bioinformatics, 2002. **18**(7): p. 908-921.
14. Casteigts, A. and P. Flocchini, *Deterministic algorithms in dynamic networks: Formal models and metrics.* 2013.
15. Juang, C.-F., *A hybrid of genetic algorithm and particle swarm optimization for recurrent network design.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004. **34**(2): p. 997-1006.
16. Lin, T.-L., et al., *An efficient job-shop scheduling algorithm based on particle swarm optimization.* Expert Systems with Applications, 2010. **37**(3): p. 2629-2636.

17. Bianchi, L., L.M. Gambardella, and M. Dorigo. *An ant colony optimization approach to the probabilistic traveling salesman problem.* in *PPSN*. 2002. Springer.
18. Del Valle, Y., et al., *Particle swarm optimization: basic concepts, variants and applications in power systems.* IEEE Transactions on evolutionary computation, 2008. **12**(2): p. 171-195.
19. Farina, M. *A neural network based generalized response surface multiobjective evolutionary algorithm*. in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. 2002. IEEE.
20. Zhou, Z., et al., *Memetic algorithm using multi-surrogates for computationally expensive optimization problems.* Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2007. **11**(10): p. 957-971.
21. Sun, X.Y., D.W. Gong, and X.P. Ma. *Directed fuzzy graph-based surrogate model-assisted interactive genetic algorithms with uncertain individual's fitness*. in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. 2009. IEEE.
22. Regis, R.G., *Particle swarm with radial basis function surrogates for expensive black-box optimization.* Journal of Computational Science, 2014. **5**(1): p. 12-23.
23. Mallipeddi, R. and M. Lee. *Surrogate model assisted ensemble differential evolution algorithm*. in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. 2012. IEEE.
24. Simpson, T.W., et al., *Kriging models for global approximation in simulation-based multidisciplinary design optimization.* AIAA journal, 2001. **39**(12): p. 2233-2241.
25. Jones, D.R., M. Schonlau, and W.J. Welch, *Efficient global optimization of expensive black-box functions.* Journal of Global optimization, 1998. **13**(4): p. 455-492.
26. Booker, A.J., et al., *A rigorous framework for optimization of expensive functions by surrogates.* Structural optimization, 1999. **17**(1): p. 1-13.
27. Parno, M.D., K.R. Fowler, and T. Hemker, *Framework for particle swarm optimization with surrogate functions.* Darmstadt Technical University, Darmstadt, Germany, 2009.
28. Shan, S. and G.G. Wang, *Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions.* Structural and Multidisciplinary Optimization, 2010. **41**(2): p. 219-241.
29. Alexandrov, N., et al., *Opportunities for breakthroughs in large-scale computational simulation and design.* 2002.
30. Pinedo, M., *Scheduling*. 2012: Springer.
31. Della Croce, F., R. Tadei, and G. Volta, *A genetic algorithm for the job shop problem.* Computers & Operations Research, 1995. **22**(1): p. 15-24.
32. Sun, L., X. Cheng, and Y. Liang, *Solving job shop scheduling problem using genetic algorithm with penalty function.* International Journal of Intelligent information processing, 2010. **1**(2): p. 65-77.
33. Lee, K.-M., T. Yamakawa, and K.-M. Lee. *A genetic algorithm for general machine scheduling problems*. in *Knowledge-Based Intelligent Electronic Systems, 1998. Proceedings KES'98. 1998 Second International Conference on*. 1998. IEEE.

34. Liu, L. and Y. Xi. *A hybrid genetic algorithm for job shop scheduling problem to minimize makespan*. in *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*. 2006. IEEE.
35. Zhou, H., W. Cheung, and L.C. Leung, *Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm.* European Journal of Operational Research, 2009. **194**(3): p. 637-649.
36. Asadzadeh, L. and K. Zamanifar, *An agent-based parallel approach for the job shop scheduling problem with genetic algorithms.* Mathematical and Computer Modelling, 2010. **52**(11): p. 1957-1965.
37. Yusof, R., et al., *Solving job shop scheduling problem using a hybrid parallel micro genetic algorithm.* Applied soft computing, 2011. **11**(8): p. 5782-5792.
38. Nik, M.A., et al., *A comparative study of metamodeling methods for the design optimization of variable stiffness composites.* Composite Structures, 2014. **107**: p. 494-501.
39. Yao, X., *A new simulated annealing algorithm.* International Journal of Computer Mathematics, 1995. **56**(3-4): p. 161-168.
40. Xinchao, Z., *Simulated annealing algorithm with adaptive neighborhood.* Applied Soft Computing, 2011. **11**(2): p. 1827-1836.
41. Ingber, L., *Adaptive simulated annealing (ASA): Lessons learned.* arXiv preprint cs/0001018, 2000.
42. Bevilacqua, A., *A methodological approach to parallel simulated annealing on an SMP system.* Journal of Parallel and Distributed Computing, 2002. **62**(10): p. 1548-1570.
43. Cordón, O., F. Moya, and C. Zarco, *A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems.* Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2002. **6**(5): p. 308-319.
44. Pajares, G. and J.M. de la Cruz, *On combining support vector machines and simulated annealing in stereovision matching.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004. **34**(4): p. 1646-1657.
45. Salcedo-Sanz, S., R. Santiago-Mozos, and C. Bousoño-Calzón, *A hybrid Hopfield network-simulated annealing approach for frequency assignment in satellite communications systems.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004. **34**(2): p. 1108-1116.
46. Wang, L. and L. Zhang, *Stochastic optimization using simulated annealing with hypothesis test.* Applied Mathematics and Computation, 2006. **174**(2): p. 1329-1342.
47. Singh, H.K., T. Ray, and W. Smith. *Surrogate assisted simulated annealing (SASA) for constrained multi-objective optimization*. in *Evolutionary Computation (CEC), 2010 IEEE Congress on*. 2010. IEEE.
48. Socha, K. and M. Dorigo, *Ant colony optimization for continuous domains.* European journal of operational research, 2008. **185**(3): p. 1155-1173.
49. Dorigo, M. and L.M. Gambardella, *Ant colony system: a cooperative learning approach to the traveling salesman problem.* IEEE Transactions on evolutionary computation, 1997. **1**(1): p. 53-66.

50. Stützle, T. and H.H. Hoos, *MAX–MIN ant system.* Future generation computer systems, 2000. **16**(8): p. 889-914.
51. Iredi, S., D. Merkle, and M. Middendorf. *Bi-criterion optimization with multi colony ant algorithms*. in *Evolutionary Multi-Criterion Optimization*. 2001. Springer.
52. Doerner, K., et al., *Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection.* Annals of operations research, 2004. **131**(1): p. 79-99.
53. Bell, J.E. and P.R. McMullen, *Ant colony optimization techniques for the vehicle routing problem.* Advanced engineering informatics, 2004. **18**(1): p. 41-48.
54. Merkle, D., M. Middendorf, and H. Schmeck, *Ant colony optimization for resource-constrained project scheduling.* IEEE transactions on evolutionary computation, 2002. **6**(4): p. 333-346.
55. Cardoso, P., M. Jesus, and A. Márquez, *MONACO-multi-objective network optimisation based on an ACO.* Proceedings of Encuentros de Geometrıa Computacional, 2003.
56. Valdez, F., P. Melin, and O. Castillo, *Modular neural networks architecture optimization with a new nature inspired method using a fuzzy combination of particle swarm optimization and genetic algorithms.* Information Sciences, 2014. **270**: p. 143-153.
57. Eberhart, R.C. and Y. Shi. *Tracking and optimizing dynamic systems with particle swarms*. in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*. 2001. IEEE.
58. Ling, S.-H., et al., *Hybrid particle swarm optimization with wavelet mutation and its industrial applications.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2008. **38**(3): p. 743-763.
59. Nickabadi, A., M.M. Ebadzadeh, and R. Safabakhsh, *A novel particle swarm optimization algorithm with adaptive inertia weight.* Applied Soft Computing, 2011. **11**(4): p. 3658-3670.
60. Wu, G., et al., *Superior solution guided particle swarm optimization combined with local search techniques.* Expert Systems with Applications, 2014. **41**(16): p. 7536-7548.
61. Wang, H., et al. *A hybrid particle swarm algorithm with Cauchy mutation*. in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. 2007. IEEE.
62. Shin, Y.-B. and E. Kita, *Search performance improvement of particle swarm optimization by second best particle information.* Applied Mathematics and Computation, 2014. **246**: p. 346-354.
63. Ratnaweera, A., S.K. Halgamuge, and H.C. Watson, *Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients.* IEEE Transactions on evolutionary computation, 2004. **8**(3): p. 240-255.
64. Zhan, Z.-H., et al., *Orthogonal learning particle swarm optimization.* IEEE transactions on evolutionary computation, 2011. **15**(6): p. 832-847.
65. Zhan, Z.-H., et al., *Adaptive particle swarm optimization.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2009. **39**(6): p. 1362-1381.
66. Das, S. and P.N. Suganthan, *Differential evolution: A survey of the state-of-the-art.* IEEE transactions on evolutionary computation, 2011. **15**(1): p. 4-31.

67. Land, A.H. and A.G. Doig, *An automatic method of solving discrete programming problems.* Econometrica: Journal of the Econometric Society, 1960: p. 497-520.
68. Clausen, J., *Branch and bound algorithms-principles and examples.* Department of Computer Science, University of Copenhagen, 1999: p. 1-30.
69. Hardy, R.L., *Multiquadric equations of topography and other irregular surfaces.* Journal of geophysical research, 1971. **76**(8): p. 1905-1915.
70. Kattan, A. and E. Galvan. *Evolving radial basis function networks via gp for estimating fitness values using surrogate models*. in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. 2012. IEEE.
71. Li, G., et al., *High-dimensional model representations generated from low order terms—lp-RS-HDMR.* Journal of computational chemistry, 2003. **24**(5): p. 647-656.
72. Wang, G.G. and S. Shan, *Review of metamodeling techniques in support of engineering design optimization.* Journal of Mechanical design, 2007. **129**(4): p. 370-380.
73. Wang, L., S. Shan, and G.G. Wang, *Mode-pursuing sampling method for global optimization on expensive black-box functions.* Engineering Optimization, 2004. **36**(4): p. 419-438.
74. Tavassoli, A., et al., *Modification of DIRECT for high-dimensional design problems.* Engineering Optimization, 2014. **46**(6): p. 810-823.
75. Gablonsky, J.M., *Modifications of the DIRECT Algorithm.* 2001.
76. Gablonsky, J.M. and C.T. Kelley, *A locally-biased form of the DIRECT algorithm.* Journal of Global Optimization, 2001. **21**(1): p. 27-37.
77. Finkel, D.E. and C. Kelley, *Convergence analysis of the DIRECT algorithm.* Optimization Online, 2004. **14**(2): p. 1-10.
78. Zhu, H. and D.B. Bogy, *Hard disc drive air bearing design: modified DIRECT algorithm and its application to slider air bearing surface optimization.* Tribology international, 2004. **37**(2): p. 193-201.
79. Huyer, W. and A. Neumaier, *Global optimization by multilevel coordinate search.* Journal of Global Optimization, 1999. **14**(4): p. 331-355.
80. Chiter, L., *DIRECT algorithm: A new definition of potentially optimal hyperrectangles.* Applied Mathematics and computation, 2006. **179**(2): p. 742-749.
81. Chiter, L., *A new sampling method in the DIRECT algorithm.* Applied Mathematics and Computation, 2006. **175**(1): p. 297-306.
82. Deng, G. and M.C. Ferris. *Extension of the direct optimization algorithm for noisy functions*. in *Simulation Conference, 2007 Winter*. 2007. IEEE.
83. Hansen, P., B. Jaumard, and S.-H. Lu, *Global optimization of univariate Lipschitz functions: I. Survey and properties.* Mathematical programming, 1992. **55**(1): p. 251-272.
84. Jie, H., Y. Wu, and J. Ding, *An adaptive metamodel-based global optimization algorithm for black-box type problems.* Engineering optimization, 2015. **47**(11): p. 1459-1480.
85. Babu, B. and S. Munawar, *Differential evolution strategies for optimal design of shell-and-tube heat exchangers.* Chemical Engineering Science, 2007. **62**(14): p. 3720-3739.

86. Kannan, S., S.M.R. Slochanal, and N.P. Padhy, *Application and comparison of metaheuristic techniques to generation expansion planning problem.* IEEE Transactions on Power Systems, 2005. **20**(1): p. 466-475.
87. Chiou, J.-P., C.-F. Chang, and C.-T. Su, *Ant direction hybrid differential evolution for solving large capacitor placement problems.* IEEE Transactions on power systems, 2004. **19**(4): p. 1794-1800.
88. Zelinka, I., G. Chen, and S. Celikovsky, *Chaos synthesis by means of evolutionary algorithms.* International Journal of Bifurcation and Chaos, 2008. **18**(04): p. 911-942.
89. Ursem, R.K. and P. Vadstrup. *Parameter identification of induction motors using differential evolution*. in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*. 2003. IEEE.
90. Jin, Y., *Surrogate-assisted evolutionary computation: Recent advances and future challenges.* Swarm and Evolutionary Computation, 2011. **1**(2): p. 61-70.
91. Islam, S.M., et al., *An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012. **42**(2): p. 482-500.
92. Kettani, O., F. Ramdani, and B. Tadili, *A Quantum Differential Evolutionary Algorithm for the Independent Set Problem.* International Journal of Computer Applications, 2012. **58**(14).
93. Regulwar, D.G., S.A. Choudhari, and P.A. Raj, *Differential evolution algorithm with application to optimal operation of multipurpose reservoir*. Journal of Water Resource and Protection, 2010. **2**(06): p. 560.
94. Ghosh, A., et al., *An improved differential evolution algorithm with fitness-based adaptation of the control parameters.* Information Sciences, 2011. **181**(18): p. 3749-3765.
95. Liu, J. and J. Lampinen, *A fuzzy adaptive differential evolution algorithm.* Soft Computing, 2005. **9**(6): p. 448-462.
96. Brest, J., et al., *Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems.* IEEE transactions on evolutionary computation, 2006. **10**(6): p. 646-657.
97. Ali, M.M. and A. Törn, *Population set-based global optimization algorithms: some modifications and numerical studies.* Computers & Operations Research, 2004. **31**(10): p. 1703-1725.
98. Zaharie, D. *Control of population diversity and adaptation in differential evolution algorithms*. in *Proc. of MENDEL*. 2003.
99. Qin, A.K., V.L. Huang, and P.N. Suganthan, *Differential evolution algorithm with strategy adaptation for global numerical optimization.* IEEE transactions on Evolutionary Computation, 2009. **13**(2): p. 398-417.
100. Omran, M.G., A. Salman, and A.P. Engelbrecht. *Self-adaptive differential evolution*. in *International Conference on Computational and Information Science*. 2005. Springer.
101. Zhang, J. and A.C. Sanderson, *JADE: adaptive differential evolution with optional external archive.* IEEE Transactions on evolutionary computation, 2009. **13**(5): p. 945-958.

102. Neri, F. and V. Tirronen, *Scale factor local search in differential evolution.* Memetic Computing, 2009. **1**(2): p. 153-171.
103. Dyn, N., D. Levin, and S. Rippa, *Numerical procedures for surface fitting of scattered data by radial functions.* SIAM Journal on Scientific and Statistical Computing, 1986. **7**(2): p. 639-659.
104. Dong, H., et al., *Multi-start Space Reduction (MSSR) surrogate-based global optimization method.* Struct. Multidiscip. Optim., 2016. **54**(4): p. 907-926.
105. Brenna, M., F. Foiadelli, and M. Longo, *Application of Genetic Algorithms for Driverless Subway Train Energy Optimization.* International Journal of Vehicular Technology, 2016. **2016**: p. 1-14.
106. Zhao, J., D. Cheng, and C. Hao, *An Improved Ant Colony Algorithm for Solving the Path Planning Problem of the Omnidirectional Mobile Vehicle.* Mathematical Problems in Engineering, 2016. **2016**: p. 1-10.
107. Garg, H., *Reliability, Availability and Maintainability Analysis of Industrial Systems Using PSO and Fuzzy Methodology.* Mapan, 2013. **29**(2): p. 115-129.
108. Baeyens, E., A. Herreros, and J. Perán, *A Direct Search Algorithm for Global Optimization.* Algorithms, 2016. **9**(2): p. 40.
109. Morrison, D.R., et al., *Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning.* Discrete Optimization, 2016. **19**: p. 79-102.
110. Xu, D. and Y. Tian, *A Comprehensive Survey of Clustering Algorithms.* Annals of Data Science, 2015. **2**(2): p. 165-193.
111. Levy, A.V. and A. Montalvo, *The Tunneling Algorithm for the Global Minimization of Functions.* SIAM Journal on Scientific and Statistical Computing, 1985. **6**(1): p. 15-29.
112. Scaria, A., K. George, and J. Sebastian, *An Artificial Bee Colony Approach for Multi-objective Job Shop Scheduling.* Procedia Technology, 2016. **25**: p. 1030-1037.
113. Ritthipakdee, A., et al., *Firefly Mating Algorithm for Continuous Optimization Problems.* Comput Intell Neurosci, 2017. **2017**: p. 8034573.
114. Garg, H., *An approach for solving constrained reliability-redundancy allocation problems using cuckoo search algorithm.* Beni-Suef University Journal of Basic and Applied Sciences, 2015. **4**(1): p. 14-25.
115. Yang, X.S. and A. Hossein Gandomi, *Bat algorithm: a novel approach for global engineering optimization.* Engineering Computations, 2012. **29**(5): p. 464-483.
116. Yang, X.-S., M. Karamanoglu, and X. He, *Flower pollination algorithm: A novel approach for multiobjective optimization*. Vol. 46. 2014.
117. Mirjalili, S., S.M. Mirjalili, and A. Lewis, *Grey Wolf Optimizer.* Advances in Engineering Software, 2014. **69**: p. 46-61.
118. Ab Wahab, M.N., S. Nefti-Meziani, and A. Atyabi, *A comprehensive review of swarm optimization algorithms.* PLoS One, 2015. **10**(5): p. e0122827.
119. Wang, L., F. Li, and J. Xing, *A hybrid artificial bee colony algorithm and pattern search method for inversion of particle size distribution from spectral extinction data.* Journal of Modern Optics, 2017. **64**(19): p. 2051-2065.
120. Gao, W.-f. and S.-y. Liu, *A modified artificial bee colony algorithm.* Computers & Operations Research, 2012. **39**(3): p. 687-697.

121. Zhu, G. and S. Kwong, *Gbest-guided artificial bee colony algorithm for numerical function optimization.* Applied Mathematics and Computation, 2010. **217**(7): p. 3166-3173.
122. Li, G., P. Niu, and X. Xiao, *Development and investigation of efficient artificial bee colony algorithm for numerical function optimization.* Applied Soft Computing, 2012. **12**(1): p. 320-332.
123. Banharnsakun, A., T. Achalakul, and B. Sirinaovakul, *The best-so-far selection in Artificial Bee Colony algorithm.* Applied Soft Computing, 2011. **11**(2): p. 2888-2901.
124. Xiang, W.-l. and M.-q. An, *An efficient and robust artificial bee colony algorithm for numerical optimization.* Computers & Operations Research, 2013. **40**(5): p. 1256-1265.
125. Li, X. and G. Yang, *Artificial bee colony algorithm with memory.* Applied Soft Computing, 2016. **41**: p. 362-372.
126. Garg, H., M. Rani, and S.P. Sharma, *An efficient two phase approach for solving reliability–redundancy allocation problem using artificial bee colony technique.* Computers & Operations Research, 2013. **40**(12): p. 2961-2969.
127. Karaboga, D. and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm*. Vol. 39. 2007. 459-471.
128. Fister, I., et al., *A comprehensive review of firefly algorithms*. Vol. 13. 2013. 34–46.
129. Yang, X.-S. and X. He, *Firefly Algorithm: Recent Advances and Applications*. Vol. 1. 2013.
130. Bhushan, B. and S.S. Pillai, *Particle Swarm Optimization and Firefly Algorithm: Performance analysis*. 2013. 746-751.
131. Mashhadi Farahani, S., B. Nasiri, and M. Meybodi, *A multiswarm based firefly algorithm in dynamic environments*. Vol. 3. 2011. 68-72.
132. Younes, M., F. Khodja, and R.L. Kherfane, *Multi-objective economic emission dispatch solution using hybrid FFA (firefly algorithm) and considering wind power penetration.* Energy, 2014. **67**: p. 595-606.
133. Talatahari, S., A.H. Gandomi, and G.J. Yun, *Optimum design of tower structures using Firefly Algorithm.* The Structural Design of Tall and Special Buildings, 2014. **23**(5): p. 350-361.
134. Hassanzadeh, T., H. Vojodi, and A.M.E. Moghadam. *An image segmentation approach based on maximum variance Intra-cluster method and Firefly algorithm*. in *2011 Seventh International Conference on Natural Computation*. 2011.
135. Jati, G.K. and Suyanto, *Evolutionary Discrete Firefly Algorithm for Travelling Salesman Problem*, in *Adaptive and Intelligent Systems: Second International Conference, ICAIS 2011, Klagenfurt, Austria, September 6-8, 2011. Proceedings*, A. Bouchachia, Editor. 2011, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 393-403.
136. Arora, S. and S. Singh, *The Firefly Optimization Algorithm: Convergence Analysis and Parameter Selection*. Vol. 69. 2013. 48-52.
137. Bidar, M. and H.R. Kanan. *Modified firefly algorithm using fuzzy tuned parameters*. in *2013 13th Iranian Conference on Fuzzy Systems (IFSC)*. 2013.

138. Gandomi, A.H., X.-S. Yang, and A.H. Alavi, *Mixed variable structural optimization using Firefly Algorithm.* Computers & Structures, 2011. **89**(23): p. 2325-2336.
139. M. Farahani, S., et al., *A Gaussian Firefly Algorithm*. Vol. 1. 2011. 448-453.
140. Yang, X.-S. and S. Deb, *Cuckoo search: recent advances and applications. Neural Computing and Applications*. Vol. 24. 2014.
141. Yatim, J., A. Zain, and N.E.N. Bazin, *Cuckoo Search Algorithm for Optimization Problems—A Literature Review and its Applications*. Vol. 28. 2014.
142. Walton, S., et al., *Modified cuckoo search: A new gradient free optimisation algorithm.* Chaos, Solitons & Fractals, 2011. **44**(9): p. 710-718.
143. Yildiz, A., *Cuckoo search algorithm for the selection of optimal machining parameters in milling operations*. Vol. 64. 2012.
144. Vazquez, R.A., *Training spiking neural models using cuckoo search algorithm.* IEEE Congress on Eovlutionary Computation (CEC'11, 2011(2011): p. 679-686.
145. Kaveh, A. and T. Bakhshpoori, *Optimum design of steel frames using Cuckoo Search algorithm with Lévy flights.* The Structural Design of Tall and Special Buildings, 2013. **22**(13): p. 1023-1036.
146. Chifu, V.R., et al., *Optimizing the Semantic Web Service Composition Process Using Cuckoo Search*, in *Intelligent Distributed Computing V: Proceedings of the 5th International Symposium on Intelligent Distributed Computing – IDC 2011, Delft, The Netherlands – October 2011*, F.M.T. Brazier, et al., Editors. 2012, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 93-102.
147. huai tein, L. and R. Ramli, *Recent Advancements of Nurse Scheduling Models and A Potential Path*. 2010.
148. Choudhary, K. and G. Purohit, *A new testing approach using cuckoo search to achieve multi-objective genetic algorithm*. Vol. 3. 2011. 117-119.
149. Bulatović, R., S. Djordjevic, and V. Djordjevic, *Cuckoo Search algorithm: A metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage*. Vol. 61. 2013. 1-13.
150. Speed, E.R., *Evolving a Mario agent using cuckoo search and softmax heuristics*. 2010. 1-7.
151. Yang, X.-S., *A New Metaheuristic Bat-Inspired Algorithm*. Vol. 284. 2010.
152. Afrabandpey, H., et al., *A Novel Bat Algorithm Based on Chaos for Optimization Tasks*. 2014.
153. Al-Betar, M., et al., *Bat-inspired Algorithms with Natural Selection mechanisms for Global optimization*. 2017.
154. Xie, J., Y.-Q. Zhou, and H. Chen, *A Novel Bat Algorithm Based on Differential Operator and Lévy Flights Trajectory*. Vol. 2013. 2013. 453812.
155. Lin, J.H., et al., *A chaotic Levy flight bat algorithm for parameter estimation in nonlinear dynamic biological systems*. Vol. 2. 2012. 56-63.
156. Yılmaz, S. and E.U. Kucuksille, *Improved Bat Algorithm (IBA) on Continuous Optimization Problems*. Vol. 1. 2013.
157. Wang, G. and L. Guo, *A Novel Hybrid Bat Algorithm with Harmony Search for Global Numerical Optimization.* Journal of Applied Mathematics, 2013. **2013**: p. 21.

158. Zhu, B., et al., *A Novel Quantum-Behaved Bat Algorithm with Mean Best Position Directed for Numerical Optimization*. Vol. 2016. 2016. 1-17.
159. Gandomi, A. and X.-S. Yang, *Chaotic bat algorithm*. Vol. 5. 2013.
160. Kielkowicz, K. and D. Grela, *Modified Bat Algorithm for Nonlinear Optimization.* International Journal of Computer Science and Network Security (IJCSNS), 2016. **16**(10): p. 46.
161. He, X., et al., *Global Convergence Analysis of the Flower Pollination Algorithm: A Discrete-Time Markov Chain Approach*. Vol. 108. 2017. 1354-1363.
162. nabil, e., *A Modified Flower Pollination Algorithm for Global Optimization*. Vol. 57. 2016.
163. Alam, D.F., D.A. Yousri, and M. Eteiba, *Flower Pollination Algorithm based solar PV parameter estimation*. Vol. 101. 2015.
164. Henawy, I., O. Abdel-Raouf, and M. Abdel-Baset, *A New Hybrid Flower Pollination Algorithm for Solving Constrained Global Optimization Problems*. Vol. 4. 2014.
165. Wang, R. and Y.-Q. Zhou, *Flower Pollination Algorithm with Dimension by Dimension Improvement*. Vol. 2014. 2014. 1-9.
166. Kanagasabai, L. and B. RavindhranathReddy, *Reduction of real power loss by using Fusion of Flower Pollination Algorithm with Particle Swarm Optimization*. Vol. 2. 2014. 97-103.
167. Kok Meng, O., et al., *Application of Modified Flower Pollination Algorithm on Mechanical Engineering Design Problem*. Vol. 165. 2017. 012032.
168. Binh, H., et al., *Improved Cuckoo Search and Chaotic Flower Pollination Optimization Algorithm for Maximizing Area Coverage in Wireless Sensor Network*. 2016.
169. Łukasik, S. and P.A. Kowalski, *Study of Flower Pollination Algorithm for Continuous Optimization*. Vol. 322. 2015. 451-459.
170. Sakib, N., et al., *A Comparative Study of Flower Pollination Algorithm and Bat Algorithm on Continuous Optimization Problems*. Vol. 7. 2014. 20-19.
171. Precup, R.-E., et al., *An Easily Understandable Grey Wolf Optimizer and Its Application to Fuzzy Controller Tuning.* Algorithms, 2017. **10**(2): p. 68.
172. Kamboj, V.K., S.K. Bath, and J.S. Dhillon, *Solution of non-convex economic load dispatch problem using Grey Wolf Optimizer.* Neural Computing and Applications, 2016. **27**(5): p. 1301-1316.
173. Emary, E., et al., *Feature Subset Selection Approach by Gray-Wolf Optimization*, in *Afro-European Conference for Industrial Advancement: Proceedings of the First International Afro-European Conference for Industrial Advancement AECIA 2014*, A. Abraham, P. Krömer, and V. Snasel, Editors. 2015, Springer International Publishing: Cham. p. 1-13.
174. Gholizadeh, S., *Optimal design of double layer grids considering nonlinear behaviour by sequential grey wolf algorithm.* International Journal of Optimimization in Civil Engineering, 2015. **5**: p. 511-523.
175. Yusof, Y. and Z. Mustaffa, *Time Series Forecasting of Energy Commodity using Grey Wolf Optimizer*. Vol. 1. 2015. 25-30.
176. Komaki, G. and V. Kayvanfar, *Grey Wolf Optimizer Algorithm for the Two-stage Assembly Flowshop Scheduling Problem with Release Time*. 2015.

177. El-Fergany, A. and H. Hasanien, *Single and Multi-objective Optimal Power Flow Using Grey Wolf Optimizer and Differential Evolution Algorithms*. Vol. 43. 2015. 1548-1559.
178. Zawbaa, H., E. Emary, and A.E. Hassanien, *Binary Grey Wolf Optimization Approaches for Feature Selection*. Vol. 172. 2016. 371-381.
179. Kohli, M. and S. Arora, *Chaotic grey wolf optimization algorithm for constrained optimization problems*. 2017.
180. Mittal, N., U. Singh, and B. Singh Sohi, *Modified Grey Wolf Optimizer for Global Engineering Optimization*. Vol. 2016. 2016. 1-16.
181. Eiben, A., et al., *Parameter Control in Evolutionary Algorithms*, in *Parameter Setting in Evolutionary Algorithms*, G.L. Fernando, F.L. Cláudio, and M. Zbigniew, Editors. 2007, Springer Verlag. p. 19-46.
182. Yang, X.S., S. Deb, and S. Fong, *Bat Algorithm is Better Than Intermittent Search Strategy.* Journal of Multiple-Valued Logic and Soft Computing, 2014. **22**(3): p. 223-237.
183. Akay, B. and D. Karaboga, *Parameter Tuning for the Artificial Bee Colony Algorithm*. Vol. 5796. 2009. 608-619.
184. Yuan-bin Mo, Y.-z.M., Qiao-yan Zheng,, *Optimal Choice of Parameters for Firefly Algorithm.* JF 2013 Fourth International Conference on Digital Manufacturing & Automation (ICDMA), 2013. **00**: p. 887-892.
185. Wang, J., B. Zhou, and S. Zhou, *An Improved Cuckoo Search Optimization Algorithm for the Problem of Chaotic Systems Parameter Estimation.* Computational Intelligence and Neuroscience, 2016. **2016**: p. 8.
186. Yang, X.-S., *Nature-Inspired Optimization Algorithms* 2014, London: Elsevier 300.
187. Rodríguez, L., O. Castillo, and J. Soria, *A Study of Parameters of the Grey Wolf Optimizer Algorithm for Dynamic Adaptation with Fuzzy Logic*, in *Nature-Inspired Design of Hybrid Intelligent Systems*, P. Melin, O. Castillo, and J. Kacprzyk, Editors. 2017, Springer International Publishing: Cham. p. 371-390.
188. Coello, C.A.C., *Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art.* Computer methods in applied mechanics and engineering, 2002. **191**(11): p. 1245-1287.
189. Garg, H., *Solving structural engineering design optimization problems using an Artificial Bee Colony algorithm*. Vol. 10. 2013. 777-794.
190. Mezura-Montes, E. and C. A Coello Coello, *A Simple Multimembered Evolution Strategy to Solve Constrained Optimization Problems*. Vol. 9. 2005. 1-17.
191. Garg, H., *A Hybrid GA-GSA Algorithm for Optimizing the Performance of an Industrial System by Utilizing Uncertain Data*. 2014. 620-654.
192. Leung, D.Y.C. and Y. Yang, *Wind energy development and its environmental impact: A review.* Renewable and Sustainable Energy Reviews, 2012. **16**(1): p. 1031-1039.
193. Jm, J., et al., *Definition of a 5MW Reference Wind Turbine for Offshore System Development.* 2009.
194. Karimi, M., et al., *A multi-objective design optimization approach for floating offshore wind turbine support structures.* Journal of Ocean Engineering and Marine Energy, 2017. **3**(1): p. 69-87.

195. Parpinelli, R. and H. S. Lopes, *New inspirations in swarm intelligence: A survey*. Vol. 3. 2011. 1-16.
196. Yang, X.-S., *Bat Algorithm: Literature Review and Applications.* Int. J. Bio-Inspired Computation, 2013. **5**(3): p. 141--149.
197. C. Bora, T., L. Coelho, and L. Lebensztajn, *Bat-Inspired Optimization Approach for the Brushless DC Wheel Motor Problem*. Vol. 48. 2012. 947-950.
198. Yang, X.-S., *Bat algorithm for multi-objective optimisation.* International Journal of Bio-Inspired Computation, 2011. **3**(5): p. 267-274.
199. Musikapun, P. and P. Pongcharoen, *Solving Multi-Stage Multi-Machine Multi-Product Scheduling Problem Using Bat Algorithm*. Vol. 35. 2012.
200. Luo, Q., et al., *Discrete Bat Algorithm for Optimal Problem of Permutation Flow Shop Scheduling*. Vol. 2014. 2014. 630280.
201. WasiUlKabir, M., et al., *A Novel Adaptive Bat Algorithm to Control Explorations and Exploitations for Continuous Optimization Problems*. Vol. 94. 2014. 15-20.
202. Fister, I., et al., *Planning the sports training sessions with the bat algorithm*. Vol. 149. 2015. 993-1002.
203. Gandomi, A.H. and X.-S. Yang, *Chaotic bat algorithm.* Journal of Computational Science, 2014. **5**(2): p. 224-232.
204. Nakamura, R., et al., *BBA: A Binary Bat Algorithm for Feature Selection*. 2012. 291-297.
205. Li, L. and Y.-Q. Zhou, *A novel complex-valued bat algorithm*. Vol. 25. 2014.
206. Roeva, O.N. and S.S. Fidanova, *Hybrid Bat Algorithm for Parameter Identification of an E. Coli Cultivation Process Model.* Biotechnology & Biotechnological Equipment, 2013. **27**(6): p. 4323-4326.
207. Hasançebi, O., T. Teke, and O. Pekcan, *A bat-inspired algorithm for structural optimization.* Computers & Structures, 2013. **128**(Supplement C): p. 77-90.
208. Hasançebi, O. and S. Kazemzadeh Azad, *Discrete size optimization of steel trusses using a refined big bang–big Crunch Algorithm*. Vol. 46. 2014. 61-83.
209. Gandomi, A., et al., *Bat algorithm for constrained optimization tasks*. Vol. in press. 2013.
210. Peres, W., et al., *Coordinated tuning of power system stabilizers using bio-inspired algorithms.* International Journal of Electrical Power & Energy Systems, 2015. **64**(Supplement C): p. 419-428.
211. Sathya, M.R. and M. Mohamed Thameem Ansari, *Load frequency control using Bat inspired algorithm based dual mode gain scheduling of PI controllers for interconnected power system.* International Journal of Electrical Power & Energy Systems, 2015. **64**(Supplement C): p. 365-374.
212. Ramesh, B., V.C.J. Mohan, and V.C.V. Reddy, *Application of bat algorithm for combined economic load and emission dispatch*. Vol. 2. 2013. 1-9.
213. Osaba, E., et al., *An Improved Discrete Bat Algorithm for Symmetric and Asymmetric Traveling Salesman Problems*. Vol. 48. 2016. 59-71.
214. Cavazzuti, M., *Optimization methods: from theory to design*, ed. 1st. 2013, -Verlag Berlin Heidelberg: Springer. 262.
215. Younis, A. and Z. Dong, *Trends, features, and tests of common and recently introduced global optimization methods.* Engineering Optimization, 2010. **42**(8): p. 691-718.

216. Forrester, A.I.J. and A.J. Keane, *Recent advances in surrogate-based optimization.* Progress in Aerospace Sciences, 2009. **45**(1): p. 50-79.
217. Gu, J., G.Y. Li, and Z. Dong, *Hybrid and adaptive meta-model-based global optimization.* Engineering Optimization, 2012. **44**(1): p. 87-104.
218. Zhang, M., W. Luo, and X. Wang, *Differential evolution with dynamic stochastic selection for constrained optimization.* Information Sciences, 2008. **178**(15): p. 3043-3074.
219. Ao, Y.-Y. and H.-Q. Chi, *An adaptive differential evolution algorithm to solve constrained optimization problems in engineering design.* Engineering, 2010. **2**(01): p. 65.
220. Hall, M., B. Buckham, and C. Crawford, *Hydrodynamics-based floating wind turbine support platform optimization: A basis function approach.* Renewable Energy, 2014. **66**: p. 559-569.
221. Saad, A.E.H., Z. Dong, and M. Karimi, *A Comparative Study on Recently-Introduced Nature-Based Global Optimization Methods in Complex Mechanical System Design.* Algorithms, 2017. **10**(4): p. 120.
222. Lee, C.-H., *WAMIT theory manual*. 1995: Massachusetts Institute of Technology, Department of Ocean Engineering.

Appendix A. List of Benchmark Test Problems

(1) *Banana function*

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \quad -2 \le x_i \le 2, \ i = 1,2$$

(2) *Zakharov Function*

$$f(x) = \sum_{i=1}^{D} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i x_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{D} i x_i\right)^4 \quad -5 \le x_i \le 10, \ i = 1,2$$

(3) *Six-hump Camel-back Function (SC),*

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 \quad -2 \le x_i \le 2, \ i = 1,2$$

(4) *Beale Function*

$$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + \left(2.625 - x_1 + x_{1x_2^3}\right)^2$$
$$-4.5 \le x_i \le 4.5, \ i = 1,2$$

(5) *Peaks Function*

$$f(x) = 3(1 - x_1)^2 e^{-x_1^2 - (x_2+1)^2} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)e^{-x_1^2} - \frac{1}{3}e^{-(x_1+1)^2 - x_2^2}$$
$$-3 \le x_1 \le 3, \quad -4 \le x_2 \le 4$$

(6) *Shubert Function*

$$f(x) = \prod_{i=1}^{D}\left(\sum_{j=1}^{5} cos((j+1)x_i + j\right) \quad -10 \le x_i \le 10, \ i = 1,2$$

(7) *Dixon and Price Function*

$$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2 \quad -10 \le x_i \le 10, \ i = 1,2, \ldots$$

(8) *Perm Function*

$$f(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{D}(j+\beta)\left(x_j^i - \frac{1}{j^i}\right)\right)^2 \quad -d \le x_i \le d, \ i = 1,2, \ldots$$

(9) *Leon Function*

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \quad -1.2 \le x_i \le 1.2, \ i = 1,2, \ldots$$

(10) *Easom Function*

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left[-(x_1-\pi)^2-(x_2-\pi)^2\right]$$

$$-100 \le x_i \le 100,\ \ i = 1,2,\ldots$$

(11) *Ackley Function*

$$f(x) = -20\,exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D}cos(cx_i)\right) + 20 + \exp(1) \qquad -100 \le x_i \le 100,\ \ i = 1,2,\ldots.$$

(12) *Levy Function*

$$f(x) = \sin^2(3\pi x_1) + (x_1-1)^2 + (x_1-1)^2[1+\sin^2(3\pi x_2)] + (x_2-1)^2[1+\sin^2(2\pi x_2)] \qquad -10 \le x_i \le 10,\ \ i = 1,2,\ldots.$$

(13) *Hartman6 function (HN6),*

$$f(\mathbf{x}) = -\sum_{i=1}^{4}\alpha_i \exp\left[-\sum_{j=1}^{6}B_{ij}\left(x_j - Q_{ij}\right)^2\right]$$

$$\alpha = \left[1, 1.2, 3, 3.2\right]^T, \quad B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$Q = 10^{-4}\begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

$$0 \le x_i \le 1,\ i = 1,2,\ldots 6$$

(14) *Sum. Squares Function*

$$f(x) = \sum_{i=1}^{D} i x_i^2 \qquad -10 \le x_i \le 10,\ \ i = 1,2,\ldots.$$

(15) *Sphere function,*

$$f(\mathbf{x}) = \sum_{i=1}^{10} x_i^2 \qquad -5.12 \le x_i \le 5.12,\ \ i = 1,2,\ldots,10.$$

(16) *16 Variable function (F16),*

$$f(\boldsymbol{x}) = \sum_{i=1}^{16}\sum_{j=1}^{16} a_{ij}\left(x_i^2 + x_i + 1\right)\left(x_j^2 + x_j + 1\right)$$

$$a_{ij(row\,1-8)} = \begin{bmatrix} 1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,1 \\ 0\,1\,1\,0\,0\,0\,1\,0\,0\,1\,0\,1\,0\,0\,0\,0 \\ 0\,0\,1\,0\,0\,0\,1\,0\,1\,1\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,1\,1\,0\,0\,0\,1\,0\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,1\,0 \end{bmatrix} \quad a_{ij(row\,9-16)} = \begin{bmatrix} 0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1 \end{bmatrix}$$

$-1 \le x_i \le 1,\ i = 1,2,\text{L}\ ,16.$

(17) *Sargan Function*

$$f(x) = \sum_{i=1}^{D} D\left(x_i^2 + 0.4\sum_{i\neq j}^{D} x_i\, x_j\right) \qquad -100 \le x_i \le 100,\ \ i = 1,2,\ldots.$$

(18) *Powell Function*

$$f(x) = \sum_{i=1}^{D=2} (x_{i-1} + 10x_i)^2 + 5(x_{i+1} - x_{i+2})^2 + (x_i - 2x_{i+1})^4 + 10(x_{i-1} - x_{i+2})^4$$

$$-4 \le x_i \le 5,\ \ i = 1,2,\ldots.$$

(19) *Schwefel Function*

$$f(x) = (\sum_{i=1}^{D} x_i^2)^{\alpha} \qquad -100 \le x_i \le 100,\ \ i = 1,2,\ldots.$$

(20) *Griewank Function*

$$f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \qquad -100 \le x_i \le 100,\ \ i = 1,2,\ldots.$$

(21) *Alpine Function*

$$f(x) = \sum_{i=1}^{D} |x_i \sin(x_i + 0.1x_i)| \qquad -5 \le x_i \le 5,\ \ i = 1,2,\ldots.$$

(22) *Egg Carte Function*

$$f(x) = x_1^2 + x_2^2 + 25\left(\sin^2(x_1) + \sin^2(x_2)\right) \qquad -100 \le x_i \le 100,\ \ i = 1,2,\ldots.$$

(23) *Rastrigin Function*

$$f(x) = 10D + \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i)] \qquad -10 \le x_i \le 10,\ \ i = 1,2,\ldots.$$

(24) *Cigar Function*

$$f(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \qquad -100 \le x_i \le 100,\ \ i = 1,2,\ldots.$$

(25) Tension/Compression Spring Design (TSD), n=3:

This problem minimizes the weight of a spring and meanwhile is subject to constraints on minimum deflection, shear stress, surge frequency, limits on outside diameter and side constraints.

$$\min \quad f(\boldsymbol{x}) = x_1^2 x_2 (x_3 + 2)$$

$$s.t. \quad g_1(\boldsymbol{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \le 0,$$

$$g_2(\boldsymbol{x}) = \frac{4x_2^2 - x_1 x_2}{12566 x_1^3 (x_2 - x_1)} + \frac{1}{5108 x_1^2} - 1 \le 0,$$

$$g_3(\boldsymbol{x}) = 1 - \frac{140.45 x_1}{x_3 x_2^2} \le 0,$$

$$g_4(\boldsymbol{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0.$$

$$0.05 \le x_1 \le 2;\ \ 0.25 \le x_2 \le 1.3;\ \ 2 \le x_3 \le 15$$

(26) Welded Beam Design (TSD), n=4:

This problem is designed for minimum cost and meanwhile is subject to constraints on shear stress, bending stress in the beam, buckling load on the bar, end deflection of the beam and side constraints.

$$\min \quad f(\boldsymbol{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2)$$

$$s.t. \quad g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{\max} \le 0,$$

$$g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{\max} \le 0,$$

$$g_3(\boldsymbol{x}) = x_1 - x_4 \le 0,$$

$$g_4(\boldsymbol{x}) = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0,$$

$$g_5(\boldsymbol{x}) = 0.125 - x_1 \le 0,$$

$$g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{\max} \le 0,$$

$$g_7(\boldsymbol{x}) = P - P_c(\boldsymbol{x}) \le 0,$$

where

$$\tau(\boldsymbol{x}) = \sqrt{(\tau')^2 + 2\tau'\tau'' \frac{x_2}{2R} + (\tau'')^2}, \quad \tau' = \frac{P}{\sqrt{2} x_1 x_2}, \quad \tau'' = \frac{MR}{J},$$

$$M = P\left(L + \frac{x_2}{2}\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}, \quad \sigma(\boldsymbol{x}) = \frac{6PL}{x_4 x_3^2},$$

$$\delta(\boldsymbol{x}) = \frac{4PL^3}{E x_3^3 x_4}, \quad J = 2\left\{\sqrt{2} x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\},$$

$$P_c(\boldsymbol{x}) = \frac{4.013 E \sqrt{x_3^2 x_4^6 / 36}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \quad G = 12 \times 10^6 \ psi,$$

$$P = 6000 \ lb, \quad L = 14 \ in, \quad \delta_{\max} = 0.25 \ in, \quad E = 30 \times 10^6 \ psi,$$

$$\tau_{\max} = 13600 \ psi, \quad \sigma_{\max} = 30000 \ psi.$$

(27) Pressure Vessel Design (PVD), n=4:

This problem is to minimize the total cost of a cylindrical vessel, including the cost of material, forming and welding. Four design variables are thickness of the pressure vessel, thickness of the head, inner radius of the vessel and length of the vessel, respectively.

$$\min \quad f(\boldsymbol{x}) = 0.6224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_1^2 x_4 + 19.84 x_1^2 x_3$$

$$s.t. \quad g_1(\boldsymbol{x}) = -x_1 + 0.0193 \le 0,$$

$$g_2(\boldsymbol{x}) = -x_2 + 0.00954 x_3 \le 0,$$

$$g_3(\boldsymbol{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0,$$

$$g_4(\boldsymbol{x}) = x_4 - 240 \le 0.$$

$$1 \times 0.0625 \le x_1, x_2 \le 99 \times 0.0625; \quad 10 \le x_3, x_4 \le 200$$

(28) Himmelblau's nonlinear optimization problem (Him), n=5:

This problem has five design variables, six nonlinear inequality constraints and ten boundary conditions.

$$\min \quad f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

$$\begin{aligned} s.t. \quad & g_1(\boldsymbol{x}) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5, \\ & g_2(\boldsymbol{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2, \\ & g_3(\boldsymbol{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4, \\ & 0 \le g_1(\boldsymbol{x}) \le 92, \ 90 \le g_2(\boldsymbol{x}) \le 110, \quad 20 \le g_3(\boldsymbol{x}) \le 25, \\ & 78 \le x_1 \le 102; \ 33 \le x_2 \le 45; \ 27 \le x_3, x_4, x_5 \le 45. \end{aligned}$$

(29) Speed Reducer Design (SRD), n=7:

This problem is to minimize the total weight of the speed reducer. It has eleven constraints which include the limits on the bending stress of the gear teeth, surface stress and transverse deflections of shafts.

$$\begin{aligned} \min \quad f(\boldsymbol{x}) = {} & 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right) \\ & + 7.4777\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) \end{aligned}$$

$$\begin{aligned} s.t. \quad & g_1(\boldsymbol{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0, \qquad g_6(\boldsymbol{x}) = \frac{\left[\left(745x_5/(x_2x_3)\right)^2 + 157.5\times 10^6\right]^{1/2}}{85x_7^3} - 1 \le 0, \\ & g_2(\boldsymbol{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0, \quad g_7(\boldsymbol{x}) = \frac{x_2x_3}{40} - 1 \le 0, \quad g_8(\boldsymbol{x}) = \frac{5x_2}{x_1} - 1 \le 0, \\ & g_3(\boldsymbol{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0, \quad g_9(\boldsymbol{x}) = \frac{x_1}{12x_2} - 1 \le 0, \quad g_{10}(\boldsymbol{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0, \\ & g_4(\boldsymbol{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0, \qquad g_{11}(\boldsymbol{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0, \end{aligned}$$

Appendix B. Kriging Formula

Assume that there are N sample points $\boldsymbol{x}^{(1)},\ \boldsymbol{x}^{(2)},\ldots,\ \boldsymbol{x}^{(N)}$ given by the true function f(x). The Kriging model realizes all the given points as follows:

$$\begin{aligned} f(\boldsymbol{x}^{(i)}) &= F(\boldsymbol{x}^{(i)}) \\ &= \mu + Z(\boldsymbol{x}^{(i)}) \end{aligned} \tag{1}$$

In the Kriging model, the parameters μ, σ^2, Θ are obtained by maximum likelihood estimation (MLE) (Forrester *et al.* 2009). Here the estimated values are given:

$$\begin{aligned} \hat{\mu} &= \frac{\mathbf{1}^{\mathrm{T}} \boldsymbol{R}^{-1} \boldsymbol{f}}{\mathbf{1}^{\mathrm{T}} \boldsymbol{R}^{-1} \mathbf{1}} \\ \hat{\sigma}^2 &= \frac{(\boldsymbol{f} - \mathbf{1}\hat{\mu})^{\mathrm{T}} \boldsymbol{R}^{-1} (\boldsymbol{f} - \mathbf{1}\hat{\mu})}{N} \\ Ln(\Theta) &= -\frac{S}{2}\ln(2\pi) - \frac{S}{2}\ln\hat{\sigma}^2 - \frac{1}{2}\ln|\boldsymbol{R}| \end{aligned} \tag{2}$$

where, $\boldsymbol{f} = [f(\boldsymbol{x}^{(1)}), f(\boldsymbol{x}^{(2)}),\ldots, f(\boldsymbol{x}^{(N)})]^{\mathrm{T}}$, R is a $N \times N$ correlations matrix whose element in the j^{th} column of the i^{th} line is defined as $R(\Theta, \boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$.

Finally, minimize the mean squared error (MSE)

$$\hat{s}^2(\boldsymbol{x}) = Var[\hat{f}(\boldsymbol{x}) - F(\boldsymbol{x})] \tag{3}$$

Meanwhile meet the following unbiasedness constraint:

$$E[\hat{f}(\boldsymbol{x})] = E[F(\boldsymbol{x})] \tag{4}$$

The best linear unbiased predictor (BLUP) $\hat{f}(x)$ results in the form as

$$\hat{f}(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{f} - \mathbf{1}\hat{\mu}) \tag{5}$$

where *r(x)* is an N-dimensional vector. The i^{th} element of *r(x)* is $R(\Theta, \boldsymbol{x}, \boldsymbol{x}^{(i)})$, and *x* is the any location to be estimated.

The final form of MSE is:

$$\hat{s}^2(\boldsymbol{x}) = \hat{\sigma}^2\left[1 - \boldsymbol{r}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}) + \frac{(1 - \mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{x}))^2}{\mathbf{1}^{\mathrm{T}}\boldsymbol{R}^{-1}\mathbf{1}}\right] \tag{6}$$