NOMADIC PEOPLE OPTIMIZER (NPO) FOR LARGE-SCALE OPTIMIZATION PROBLEMS

SINAN QAHTAN MOHAMD SALIH

DOCTOR OF PHILOSOPHY

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name	: Sinan Qahtan Mohamd Salih
Date of Birth	: 21/11/1988
Title	: Nomadic People Optimizer for Large-Scale Optimization Problems
Academic Session	: Semester 2 2018/2019

I declare that this thesis is classified as:

☐ CONFIDENTIAL	(Contains confidential information under the Official Secret Act 1997)*
☐ RESTRICTED	(Contains restricted information as specified by the organization where research was done)*
☑ OPEN ACCESS	I agree that my thesis to be published as online open access (Full Text)

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

(Student's Signature)	(Supervisor's Signature)
A12762618	Dr. AbdulRahman A. Alsewari
New IC/Passport Number	Name of Supervisor
Date: 18 FEB 2018	Date: 18 FEB 2018

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

SUPERVISOR'S DECLARATION

We hereby declare that we have checked this thesis and in our opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Doctor of Philosophy.

____

(Supervisor's Signature)

Full Name	:	DR. ABDULRAHMAN A. ALSEWARI
Position	:	SENIOR LECTURER
Date	:	18 FEB 2018

____

(Co-supervisor's Signature)

Full Name	:	DR. MOHAMAD FADLI ZOLKIPLI
Position	:	SENIOR LECTURER
Date	:	18 FEB 2018

STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

____

(Student's Signature)

Full Name	: SINAN QAHTAN MOHAMD SALIH
ID Number	: PCC15005
Date	: 18 FEB 2018

NOMADIC PEOPLE OPTIMIZER (NPO) FOR LARGE-SCALE OPTIMIZATION PROBLEMS

SINAN QAHTAN MOHAMD SALIH

Thesis submitted in fulfillment of the requirements
for the award of the Doctor of Philosophy
Degree in Computer Science

Faculty of Computer Systems & Software Engineering (FSKKP)

UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2019

ACKNOWLEDGEMENTS

My sincere gratefulness and love to my housewife (Saba), my lovely son (Taym), and my daughter (Tara). For their endless love, prayers, sacrifice and support to pursue and successfully complete my studies.

To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my parents, who supported me emotionally. I always knew that you believed in me and wanted the best for me. Thank you for teaching me that my job in life was to learn, to be happy, and to know and understand myself; only then could I know and understand others.

Foremost, I would like to express my sincere gratitude to my supervisor Dr. AbdulRahman A. Alsewari, and my Co-supervisor Dr. Mohamad Fadli Zolkipli for taking out time to ensure qualitative supervision of this research. Thanks a lot for your support and frequent feedback.

ABSTRAK

Beberapa dekad kebelakangan ini penyelidik telah menggunakan beberapa metodologi yang diilhamkan daripada masalah pengoptimuman yang kompleks. Kaedah carian berketentuan klasik diketahui sering terperangkap dalam minimum tempatan dan berprestasi kurang baik bagi masalah dimensi yang tinggi. Masalah lengkap-NP dianggap sebagai masalah pengoptimuman global yang umum; oleh itu, terdapat keperluan sumber pengkomputeran yang terlalu tinggi untuk memastikan pengiraan jitu bagi minimum global. Metaheuristik ditakrifkan sebagai satu proses generasi lelaran yang memberi petunjuk kepada sesuatu heuristik bawahan melalui gabungan konsep pintar yang berbeza untuk meneroka dan mengeksploitasi ruang penyelesaian; mereka menggunakan strategi pembelajaran untuk menstrukturkan maklumat dalam usaha menubuhkan penyelesaian hampir-optimum yang efisien. Tiga masalah besar dihadapi ketika merancang metaheuristik; masalah pertama ialah mengimbangi penerokaan dengan keupayaan eksploitasi (yang membawa kepada penumpuan pramatang atau memerangkap di dalam minimum tempatan), manakala masalah kedua adalah pergantungan algoritma tersebut kepada parameter mengawal, yang merupakan parameter dengan nilai-nilai optimum yang tidak diketahui. Masalah terakhir adalah keupayaan algoritma itu untuk menyelesaikan masalah skala-besar, yang kebanyakannya terdiri dari masalah dunia sebenar. Dalam tesis ini, suatu metaheuristik baru yang diilhamkan oleh alam semula jadi yang dipanggil "Pengoptimum Orang Nomad (NPO)" telah direka. NPO diilhamkan oleh gaya hidup nomad. Algoritma yang dicadangkan mensimulasikan tingkah laku nomad apabila mereka mencari sumber kehidupan (air atau padang ragut). Komponen asas algoritma tersebut terdiri daripada beberapa puak dan setiap puak mencari tempat yang terbaik (atau penyelesaian yang terbaik) berdasarkan kedudukan pemimpin mereka. Interaksi antara puak ini diilhamkan oleh konsep kumpulan (-kumpulan) orang yang dikuasai oleh pemimpin (-pemimpin) mereka. Para pemimpin puak secara berkala bertemu di dalam bilik untuk memilih pemimpin terbaik keseluruhan yang mempunyai kawalan ke atas semua pemimpin yang lain. "Pendekatan Bilik Mesyuarat (MRA)" ini memastikan keseimbangan antara keupayaan penerokaan dan eksploitasi NPO yang dicadangkan. NPO tersebut telah diuji dan dinilai berdasarkan enam puluh fungsi ujian tidak dikekang penanda aras. Tambahan lagi, kebolehskalaan NPO itu dinilai secara menyelesaikan lapan belas masalah berskala-besar. Keputusan eksperimen mengesahkan bahawa NPO yang dicadangkan berprestasi lebih baik daripada beberapa metaheuristik baru-baru ini dari segi mencapai penyelesaian terbaik, kebolehskalaan, kerumitan masa, dan kadar penumpuan. NPO ini berjaya menyelesaikan 52 daripada 60 fungsi ujian bersaiz normal manakala 16 dari 18 masalah berskala-besar telah sama-sama diselesaikan. Prestasi yang baik juga dicapai dengan NPO berkenaan dari segi hingar dan masalah maklumat yang terhad. Suatu ujian Wilcoxon Signed-Rank dilakukan untuk mengukur prestasi statistik pasangan bagi algoritma berkenaan dan daripada keputusan, NPO merekodkan prestasi statistik yang lebih baik berbanding dengan algoritma penanda aras yang lain. Dengan tegas boleh dinyatakan bahawa, penilaian eksperimen dan statistik yang dilakukan dalam kajian ini telah membuktikan keupayaan NPO yang dibangunkan ini untuk menyelesaikan masalah pengoptimuman dunia-sebenar.

ABSTRACT

Researchers have in the past few decades resorted to several methods that are inspired from complex optimization problems. The classical deterministic search methods are known to often get trapped in local minimum and do perform poorly on high dimensional problems. A metaheuristic is defined as an iterative generation process which guides a subordinate heuristic through a combination of different intelligent concepts for exploring and exploiting the solution space; they employ learning strategies to structure information in order to establish efficient near-optimal solutions. Three major problems are encountered when designing metaheuristics; the first problem is balancing exploration with exploitation capabilities (which leads to premature convergence or trapping in the local minima), while the second problem is the dependency of the algorithm on the controlling parameters, which are parameters with unknown optimal values. The final problem is the ability of the algorithm to solve large-scale problems, which mostly are the real world problems. In this thesis, a novel nature-inspired metaheuristic called "Nomadic People Optimizer (NPO)" was designed. The NPO is inspired by the lifestyle of the nomads. The proposed algorithm simulates the behavior of the nomads when they are searching for life sources (water or grazing fields). The basic component of the algorithm consists of several clans and each clan searches for the best place (or best solution) based on the position of their leader. The interaction between these clans is inspired by the concept of a group(s) of people controlled by their leader(s). The leaders of the clans periodically meet in a room to select an overall best leader who has control over all the other leaders. This "Meeting Room Approach (MRA)" ensures a balance between the exploration and exploitation capabilities of the proposed NPO. NPO provides two steps for exploitation part, while the exploration is performed using another step. The local search of NPO is implemented using a unique distribution formula, while the global search ability contains a levy flight equation which generates a step for moving the families towards the new positions. The NPO was tested and evaluated based on sixty unconstrained benchmark test functions. Additionally, the scalability of the NPO was evaluated by solving eighteen large-scale problems. The experimental results confirmed that the proposed NPO performed better than some of the recent metaheuristics in terms of achieving the best solutions, scalability, time complexity, and convergence rate. The NPO successfully solved 52 out of 60 (86.6%) normal sized test functions while 16 out of 18 (88.8%) large-scale problems were equally solved. Good performances were also achieved with the NPO with respect to noise and limited information problems. A Wilcoxon Signed-Rank Test was performed to measure the pair-wise statistical performances of the algorithms and from the results, NPO recorded a better statistical performance compared to the other benchmarking algorithms. Conclusively, the experimental and statistical evaluations performed in this study proved the capability of the developed NPO in solving real-world optimization problems.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

f_i	Objective function
X_i	The variables of the problem
Max	Maximizing Problem
Min	Minimizing Problem
σ	Sheikh of the Clan
$\overrightarrow{\sigma_c}$	The position of the Sheikh
UB	Upper Bound
LB	Lower Bound
$\overrightarrow{x_i}$	The position of a family
$Rand$	Random value between 0 and 1
Rd	Reduce of the circle
θ	The value of the angle
Ψ	The direction variable
a_c	Area of the clan

LIST OF ABBREVIATIONS

SBA	Swarm-Based Algorithms
PBA	Physics-Based Algorithms
EA	Evolutionary Algorithms
GA	Genetic Algorithms
GP	Genetic Programming
ES	Evolutionary Strategies
PSO	Particle Swarm Optimization
ABC	Artificial Bees Colony
FFA	Firefly Algorithm
BA	Bat Algorithm
GWO	Grey Wolf Optimizer
BFO	Bacterial Foraging Optimization
CSA	Cuckoo Search Algorithm
FSO	Fish Swarm Optimization
ABO	African Buffalo Algorithm
TLBO	Teaching-Learning-Based Optimizer
SELO	Socio-Evolution & Learning Optimizer
CEA	Cultural Evolution Algorithm
WC	Water Cycle
SA	Simulated Annealing
GSA	Gravitational Search Algorithms
MB	Mine Blast
NPO	Nomadic People Optimizer
MRA	Meeting Room Approach
U-N	Unimodal None Separable
U-S	Unimodal Separable
M-N	Multimodal None Separable
M-S	Multimodal Separable
NIAs	Nature-Inspired Algorithms
NNIAs	Non-Nature-Inspired Algorithms
SOM	Single Objective Metaheuristics
MOM	Multi-Objective Metaheuristics

CHAPTER 1

INTRODUCTION

1.1 Background

One of the main areas of emphasis in industrial engineering is optimization. In optimization problems, one seeks to minimize or maximize some objective function's value while simultaneously satisfying some set of specific constraints. Practically motivated optimization problems are usually so difficult and contain so many decision variables that solving them by hand is almost impossible in many cases. This fact encourages the use of computer-based algorithms to analyse optimization problems. For many problems, there are straight-forward and easy-to-understand algorithms that produce either optimal or near-optimal solutions. However, many of these algorithms can be so computationally intensive that using them would not be reasonable to obtain solutions in a reasonable, practically acceptable amount of time.

Optimization problems can be divided into two main classes: those in the class Deterministic Polynomial (P) and those in the class Non-Deterministic Polynomial (NP). There are algorithms that are able to solve problems in class P optimally in an amount of time that is bounded by a polynomial function of the problem's size. On the other hand, there are no known polynomial-time algorithms available to solve problems in class NP (i.e., non-polynomial). As producing effective solutions to class NP problems is important in practice, as many problems of practical interest are in class NP, these challenging problems have been the focus of much previous and current research. Optimization problems can be classified in general into two main classes, Continuous Problems and Discrete Problems based on the decision variables. .

Optimization processes involve a holistic search for the optimal response to a given problem, as encountered in different fields. New optimization algorithms are primarily developed for the establishment of optimal solutions to optimization problems in such a way that the given quantity is optimized based on a given set of constraints (Engelbrecht, 2007; Lynn and Suganthan, 2017).

This definition, being a simple definition of optimization, conceals several complex issues such as a) different types of data may be combined in a given solution; (b) the search area may be restricted by nonlinear constraints; (c) the convolution of the solution space with several individual solutions, (d) the tendency of the features of the problem changing with time; and (e) the presence of conflicting objectives in the optimized quantity. These are some problems that portray the complexities that could be encountered by an optimization algorithm.

When searching for a solution to a problem in a high-dimensional solution space, classical optimization algorithms may not suitably achieve accurate solutions due to the exponential increase in the search space with the problem size. It is, therefore, not feasible to solve high-dimensional search space optimization problems using exact techniques such as exhaustive search (Rashedi et al., 2009). Another problem of the classic optimization algorithms is their inability to find sufficient global optima (local optima stagnation). Furthermore, some of the classical optimization algorithms need search space derivation as well. It is, therefore, pertinent that these classical algorithms may not adequately solve real-world optimization problems (Beyer et al., 2014; Mousavirad and Ebrahimpour-Komleh, 2017).

Metaheuristic algorithms are currently being used as the primary approach to achieving optimal solutions to real optimization issues (Boussaïd et al., 2013). These approaches mainly benefit from the stochastic operators which differentiates them from the deterministic algorithms (Bonabeau et al., 1999) which reliably establishes the solution to a given problem using similar starting points. This behavior, however, leads to entrapment in the local optima which is regarded as a major problem of the deterministic approach. Local optima stagnation refers to ability of an algorithm to find just the local solutions to a problem and consequently failing to find the true global solution (optimum). Since there are many local solutions to real problems, it may be difficult to reliably establish the global optimum using deterministic algorithms (Gupta and Deep, 2018; Mirjalili, 2015).

The metaheuristic algorithms may be classified using many criteria, and this may be illustrated by their classification based on their features with respect to their search path, memory usage, type of deployed neighborhood exploration, as well as the magnitude of the solutions carried from one iteration to the next. Based on the available literature, metaheuristics are basically grouped into population-based metaheuristics (PBMs) and single-solution based metaheuristics (SBMs). Generally, the SBMs are exploitative-oriented while the PBMs are more explorative-oriented.

A metaheuristic approach can only successfully optimize a given problem if the right balance between exploration (diversification) and exploitation (intensification) is established. Exploitation is necessary for identifying the search parts that have quality solutions, and also important for the intensification of the search in the potential accumulated search spaces. The existing metaheuristic algorithms differ in the way they try to balance exploration with exploitation (Birattari et al., 2001; X. S. Yang et al., 2017).

This work proposed a new nature-inspired framework, which is based on the movement of nomads when searching for the sources of food for their herd in the desert. The proposed algorithm, known as 'Nomadic People Optimizer (NPO)' is a multi-swarm metaheuristic which contains a novel cooperative approach that enhances the interaction between the swarms. The proposed multi-swarm approach, known as 'Meeting Room Approach (MRA)' enhances the ability of NPO to balance exploration with exploration.

1.2 Problem Statement

Several algorithms have been developed over the years for solving different optimization problems. Majority of these algorithms are dependent on the nonlinear and numerical linear programming approaches which requires extensive gradient information and often strive to enhance the solution around the starting point. The numerical-based optimization algorithms are useful in achieving the optimal global solution for ideal and simple models. However, various real-world engineering problems are complicated and difficult. In the presence of more than one local optimum in a given optimization problem, the outcome may depend on selecting a starting point that will offer an optimal solution which may not represent the global best. Furthermore, an unstable gradient search may be achieved when the constraints and objective function have sharp or multiple peaks (Kallioras et al., 2018; Siddique and Adeli, 2015; Slowik and Kwasnicka, 2018). Due to the problems relating to the efficiency and accuracy of the current numerical methods, many studies have relied on the simulation and nature-inspired metaheuristics to solve most engineering optimization problems (X. S. Yang et al., 2016).

In optimization, metaheuristics are a group of algorithmic frameworks that are inspired by natural phenomena. They establish optimal solutions to optimization problems via a synergistic combination of certain rules and degrees of randomness. As such, the metaheuristics can be applied in solving several optimization problems and still experience minute changes in their overall algorithmic framework. Metaheuristics can be effectively deployed in solving real-world optimization problems because they are simple to design (naturally inspired), and have less specificity and high problem-solving potential (Al-Dabbagh et al., 2018).

The existing literature suggests the effectiveness of the metaheuristics in solving several design problems and points towards their ability to solve highly complex NP-hard problems searching (Akay and Karaboga, 2012; Črepinšek et al., 2013; Kashif et al., 2018; Silberholz and Golden, 2010; X. S. Yang et al., 2017). However, there is still lack of studies focusing on large-scale multidimensional problems.

Furthermore, tuning of control parameters can also a relevant issues as far as the existing metaheuristic is concerned. To be specific, the tuning process can be painstakingly difficult even for a small dimension problem, let alone for dealing with large-scale multidimensional problems. Specifically, poor tuning of the control parameters leads to inefficient exploration and exploitation, hence, affecting the performance of the metaheuristic algorithm at hand. Therefore, a parameter-free metaheuristic is well desired in terms of reducing the complexity of parameter tuning, and can be used in different domains without any additional adaptive methods.

To overcome the above mentioned drawbacks in the existing metaheuristics, a novel parameter-free multi-swarm metaheuristic is proposed in this thesis. The proposed algorithm with its unique structure has the ability to handle the large-scale problems.

1.3 Research Objectives

The main aim of this thesis is to offer a new swarm-based solution for the optimization field in general, and metaheuristics in particular. The solution can be implemented for solving (large scale) different types of optimization problems. To achieve this aim, the objectives of this study are:

i) To propose a novel parameter-free multi-swarm metaheuristic inspired by the movement of nomads when searching for the sources of food in the desert. The proposed algorithm, known as Nomadic People Optimizer (NPO) algorithm.

ii) To implement NPO algorithm for solving the unconstrained global optimization problems.

iii) To evaluate and test the performance of the NPO algorithm in terms of best solutions, balancing between exploration and exploitation, convergence analysis, scalability and time complexity on a combination of normal and large-scale unimodal and multimodal test functions.

1.4 Research Scope

This study mainly focuses on the development of an efficient swarm-based metaheuristic for solving global optimization problems. The proposed metaheuristic is a new nature-inspired algorithm that mimics the migration of nomads when searching for foods and water sources in the desert. In addition, it uses a novel multi-swarm approach for balancing its exploration and exploitation abilities. The proposed metaheuristic is validated based on 60 selected continuous test functions. Additionally, it is evaluated based on its ability to solve large-scale problems, as well as the convergence analysis. The comparison stage is done by benchmarking against five well-known swarm intelligence metaheuristics comprising of Particle Swarm Optimization (PSO), Artificial Bees Colony (ABC), Flower Pollination Algorithm (FPA), Grey Wolf Optimizer (GWO), and Firefly Algorithm (FA) respectively. Each metaheuristic was selected based three main characteristics, the source of inspiration, the type of the algorithm, and the structure of the algorithm.

1.5 Research Limitations

The study does not cover other forms of optimization issues such as combinatorial problems. Furthermore, the evolution-based algorithms such Genetic Algorithms (GA) and Genetic Programming (GP), as well as the other single solution-based algorithms like Simulated Annealing (SA) and Tabu Search (TS) algorithms are not considered as well, because they are not swarm based metaheuristics. Additionally, the hyper-heuristic is not covered in this study due to two main reasons, first, it is a different approach consists of several heuristics for solving specific optimization problem. While the second reason is that the heuristics used in the hyper-heuristic model face the same challenges in the metaheuristics.

1.6 Thesis Organization

This thesis is presented in five chapters to facilitate easy reading and understanding.

The current chapter provides an overview of the problem statement, the objectives, scope, and limitations of the study.

Chapter two presents a theoretical background to the field of optimization. This chapter is divided into three main sections; the first sections provides a review of the optimization processes, its problems, components, and general structure. The second and third sections reviewed metaheuristics and nature-inspired algorithms. The state-of-the-art in swarm intelligence metaheuristics was also reviewed in this chapter.

Chapter three describes the overall methodology followed to achieve the research objectives. It is introduced in a general operational framework that contains all phases and steps needed to be conducted in this work.

Chapter four explains the formulation of the proposed metaheuristic called "Nomadic People Optimizer (NPO)". This chapter also presents a deep analysis of the NPO in terms of its exploration and exploitation. Additionally, the Meeting Room Approach (MRA) and the five operators of the NPO were also explained in this chapter.

Chapter five describes the performance of the NPO based on a combination of benchmarking test functions. This benchmarking test set contains 60 test functions which are classified into Unimodal-None Separable (U-N), Unimodal-Separable (U-S), Multimodal-None separable (M-N), and Multimodal-Separable (M-S). The chapter also illustrated the experimental settings. The final section of this chapter discusses the results of the NPO execution on the benchmarking test sets.

The final chapter provides a summary of the research, as well as the conclusions drawn from the study. Suggestions for future works are also presented in this chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

There are several optimization problems in different fields. These problems can be discrete, linear, continuous, nonlinear, non-convex, or non-smooth in nature. Several conventional methods, such as gradient-based methods can be used to handle the continuously differentiable problems, but complex problems (like non-differentiable and non-convex problems) may not be solved efficiently with some conventional methods. Irrespective of the several existing methods for handling complex optimization problems, it is still difficult to achieve optimal results without investing much computational cost and effort.

Researchers have continuously relied on natural phenomena for learning and designing novel metaheuristics. In the literature, many nature-inspired algorithms which are based on the natural selection and evolution of biological systems have been proposed. Despite proposing several algorithms for dealing with several optimization applications, there is still no single algorithm which can solve all optimization problems. This is the basis for the continued effort in the development of more efficient algorithms.

In this chapter, a review of the relevant literature was carried out to explore the existing gaps to be filled in this work. The review started with the investigation of the optimization concept and tracing of the development of optimization algorithms. Furthermore, the various kinds of existing optimization algorithms in the literature were reviewed, followed by the examination of some common stochastic algorithms in terms of their strengths, weaknesses, and application. An exhaustive classification/taxonomy of the optimization algorithms concluded this chapter. An illustration of the review process carried out in this chapter is depicted in Figure 2.1.

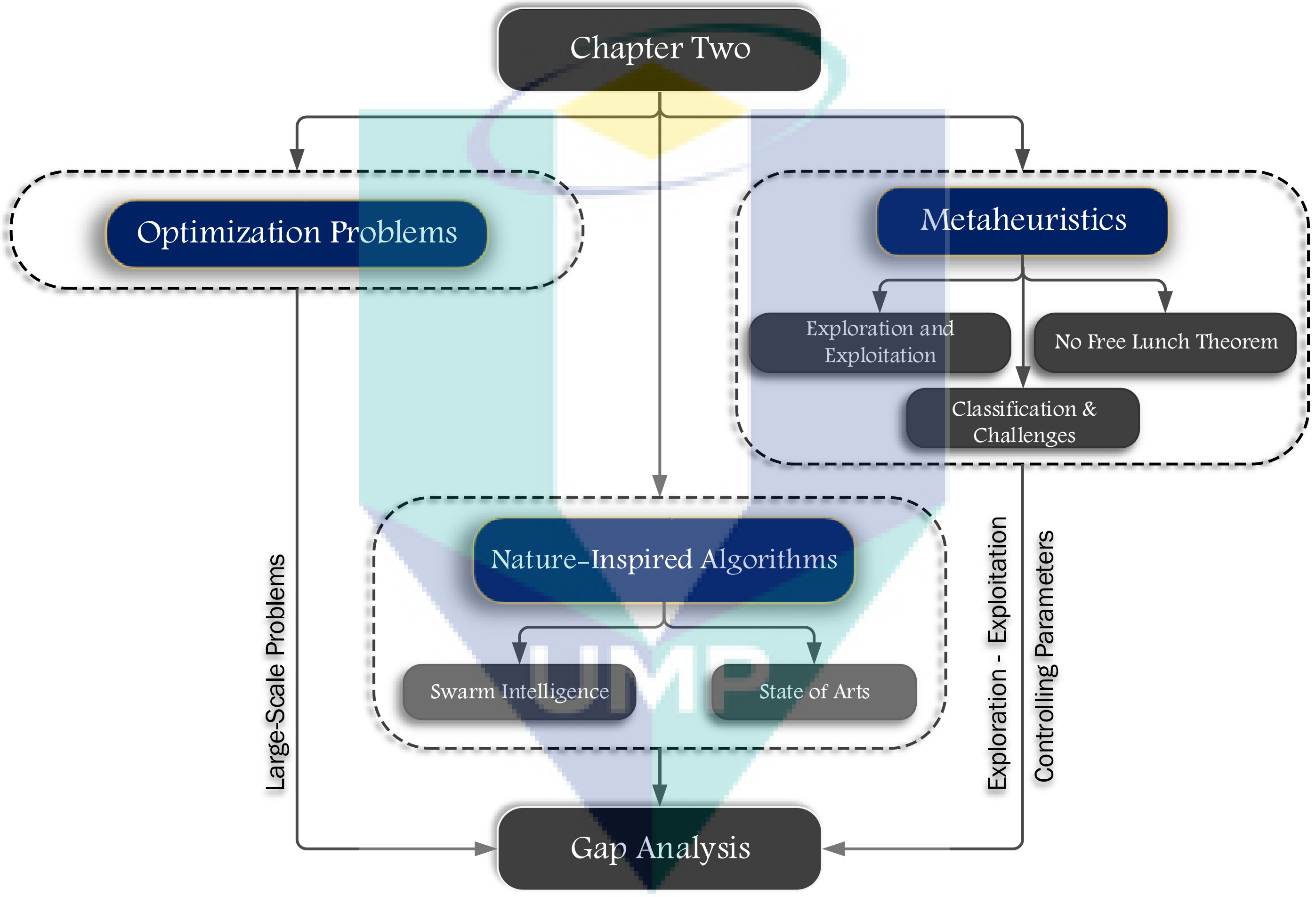


Figure 2.1 Main concepts covered in Chapter Two

2.2 Optimization Problems

These are problems which while seeking to either minimize or maximize the mathematical function of several variables with respect to certain constraints, form a unique class of problems. Several problems (theoretical and real-world) can be modeled in this general framework. Generally, the structure of mathematical models (or mathematical programming model) can be represented thus(Koziel and Yang, 2011; Mirjalili, 2016; Weise et al., 2009):

$$Max\ or\ Min\ f_i(x), \qquad (i = 1, 2, 3, \ldots, M) \qquad 2.1$$
$$subject\ to\ \ h_j(x) = 0, \qquad (j = 1, 2, 3, \ldots, J), \qquad 2.2$$
$$g_k(x) \le 0, \qquad (k = 1, 2, 3, \ldots, K) \qquad 2.3$$

where x represents the decision variables, and $f_i(x), h_j(x),$ and $g_k(x)$ are functions of the design vector:

$$\boldsymbol{x} = \ (x_1, x_2, x_3, \ldots, x_n)^T \qquad 2.4$$

Therefore, each optimization problem is composed of the following(Engelbrecht, 2007; Talbi, 2009):

- **An objective function (OF):** The OF is a representation of the quantity that needs to be optimized (i.e., the exact quantity that needs minimization or maximization). Sometimes, the objective function of most problems, especially constraint-satisfaction problems (CSP), are not defined, rather, the major objective finding a solution that will satisfy a set of given constraints.
- **A set of unknown variables:** These variables have an influence on the value of the OF. Let x represent the unknowns (also called the independent variables), then, $f(x)$ depicts the quality of the solution x.
- **A set of constraints**: These constraints restrict the values to be assigned to the unknowns. In most problems, the boundary constraints which limits the range of values for each variable are defined. The constraints can be more complex and exclude certain sets of solutions from being regarded as solutions. Figure 2.2 shows the architecture of an optimization problem.

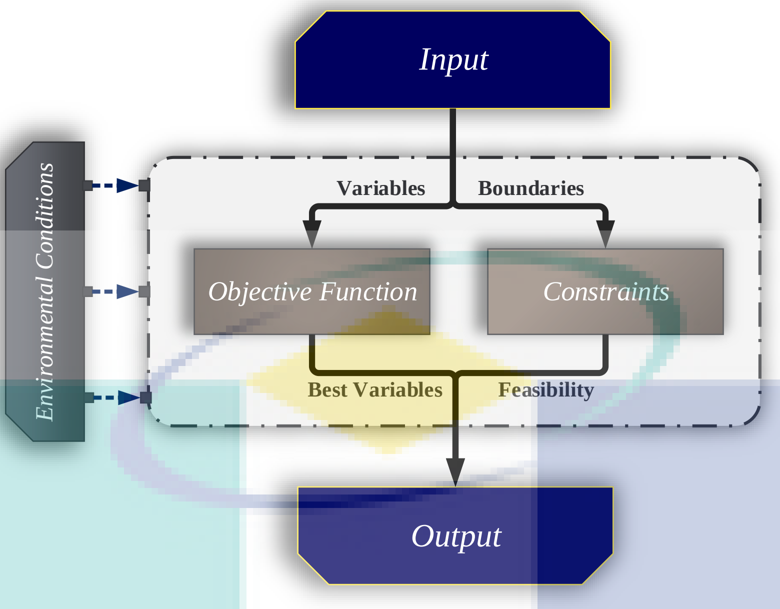


Figure 2.2 The architecture of an optimization problem

The number of variables which influences the objective function is one of the main components in any optimization problem. Optimizations problems which have only one variable to be optimized are called univariate problems, but when there are more than one optimizable variables, they are called multivariate problems. The multivariate problems are of two types; the first type has a fixed number of variables, while the second type has a non-fixed number of variables. Most studies in the literature have focused on optimization problems with about 50 or less number of variables, and when compared to real-world problems, the observed dimensionality is significantly low. Meanwhile, the existence and scalability of any algorithm that can be directly used in large-scale problems are yet to be verified. It is recommended to test optimization problems with about 50 to thousands of variables.

Problem solutions established by using optimization algorithms are classified based on their quality. These solutions can be classified as either local or global optima. The best set of solutions achieved are called the global optima (or minimum), as shown in Figure 2.3 for a minimization problem. Some problems sometimes may present more than one global optima.

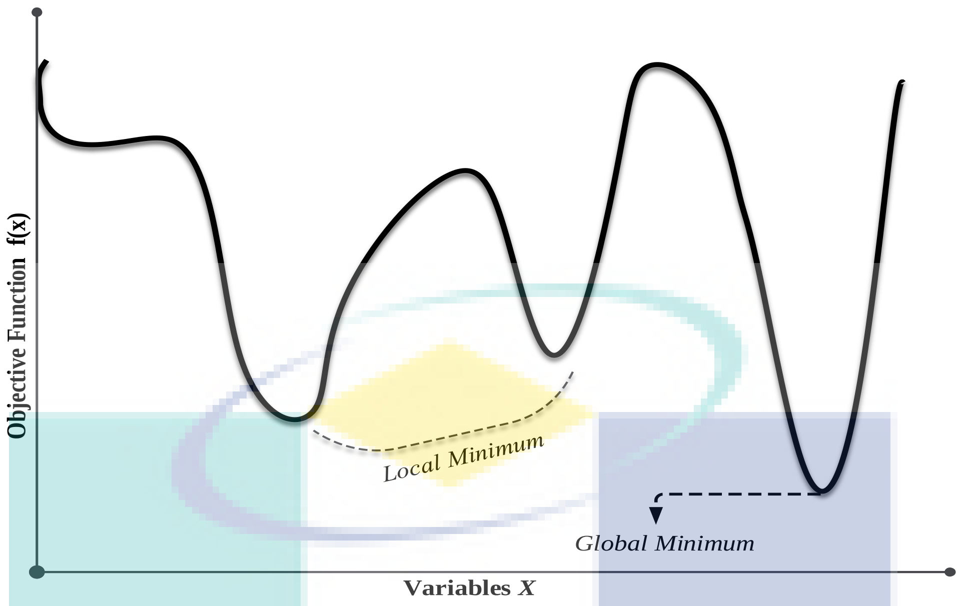


Figure 2.3 Types of optima

2.3 Metaheuristics

Optimization is a universal concept which has found application in several fields. Optimization techniques are used to intelligently solve problems by opting for the optimum from a pool of available sources. Metaheuristics have been commonly used in the field of optimization compared to other methods due to the simplicity and robustness of their outcome when used in several fields. Several studies have been conducted in the area of metaheuristics, including the introduction of novel methods, performance evaluations and applications (Abdel-Basset et al., 2018; Hussain et al., 2018; Mortazavi et al., 2018). Meanwhile, it is still believed that the field of metaheuristics is yet to mature compared to mathematics, physics, or chemistry (Sörensen et al., 2018). With time, several studies are anticipated on the issues facing metaheuristic computing.

Metaheuristics provide solutions to optimization problems by searching for the best approach to a given problem (Gendreau and Potvin, 2018). This solution search can be executed using several agents which actually form a pool of emerging solutions during multiple iterations based on a set of mathematical principles or equations. The iterations are constantly executed until a solution that meets most of the set criteria is found. This optimum solution is considered as the best achieved solution, and such a system is said to have converged (Yang and Deb, 2014). On the contrary, heuristic methods attain near-optimal solutions with less computation time, but they mostly depend on the problem.

The term "*meta*" in metaheuristics signifies the superiority of metaheuristics to the heuristics. The metaheuristics have recorded great application success and are likely to offer solutions at affordable computational costs. Classical metaheuristics can be hybridized with good heuristics to achieve better results for most real-world issues. To establish a theoretical basis for metaheuristics, it is necessary to analyse the basic metaheuristic computing terms with adaptive intelligent behaviours. The following definitions have been provided by Wang (2010) to serve this purpose:

Definition 1. Heuristics are problem-solving approaches that employs trial and error method to find the solution to problems.

Definition 2. Metaheuristics are higher-level heuristics which deploys a more general approach when solving optimization problems.

Definition 3. Metaheuristic computing refers to an aspect of adaptive computing that involves the application of general heuristic principles in addressing a range of computational tasks.

Based on these definitions, metaheuristics can be defined as depicted in the following mathematical formulation (Wang 2010):

Definition 4 A metaheuristic (MH) can be represented as:

$$MH = (O, A, R^C, R^i, R^o) \qquad 2.5$$

where O represents a set of metaheuristic approaches, A *represents a* set of generic algorithms; $R^C = O \times A$ represents a collection of internal relations; $R^i \subseteq A' \times A$, A' denotes a collection of input relations; C' represents a collection of external concepts; and c is the environment concept. $R^i = A' \times A$ may conveniently be represented as $R^i = C' \times c$. $R^O \subseteq c \times C'$ represents a collection of output relations.

Apart from the mentioned concepts, several key factors also need to be considered, such as diversification or exploration, local versus global minima, intensification or exploitation, neighbourhood search, avoiding local minima, evolutionary computing, local versus global search, as well as swarm intelligence. Additionally, there are other basic metaheuristics strategies such as exploration-exploitation balancing, searching for a potential or promising neighbour, avoiding efficient or inappropriate neighbours, and keeping from searching unpromising neighbours.

2.3.1 Exploration and Exploitation

Metaheuristics are considered as an efficient approach to achieving acceptable solutions to a complex problem through trial and error in a reasonable computational time. The limit of the solution search depends on the complexity of the problem to be addressed, but the aim is to find the best solution within an acceptable time frame. It is not certain that a given approach can yield the best solutions; it also not known whether a chosen algorithm will work even though the basic components that might make it work may be known. The aim is to have an efficient and reliable algorithm which can give quality solutions at any time. it is expected that among the obtained quality solutions, some may be near optimal even though there is often no assurance for such optimality (Abdel-Basset et al., 2018; X. S. Yang, 2013; X. S. Yang et al., 2016).

However, most typical metaheuristics can have global convergence, and as such, can find the global optimum within a few numbers of iterations. Therefore, metaheuristics are perfectly suited for solving global optimization problems. The Cuckoo search, for instance, uses both search (with good convergence) and randomization techniques (with highly efficient Levy flights) (Yang and Deb, 2009a). For metaheuristics to be efficient, they must have specialized attributes, and one of such attributes is an ability to find new solutions which can improve either the existing or previous solutions. It must also establish global optimum in the related search areas. Furthermore, it must be able to escape local optima and shouldn't be trapped in a local mode. Achieving such efficiency requires a proper combination of the necessary parameters under appropriate conditions; often, it requires a proper exploration-exploitation balancing. Meanwhile, this exploration-exploitation balancing is an optimization issue that is yet to be resolved (Yang, 2010b; Yang et al., 2017).

In the modern metaheuristics, an important component is exploration or diversification. It is often used by randomization and makes an algorithm able to escape local optimum in order to explore globally (Blum and Roli, 2003; Yang, 2010). Randomization can as well be deployed for a local search around the 'current best' if steps are locally restricted to the region. Randomization can search the solution space on a global scale only when the steps are large. The performance of any metaheuristic can be controlled by fine-tuning the right amount of its randomness and balancing its global and local searches. Randomization techniques are simple methods when using Gaussian or uniform distributions, but can be more complex when used in Monte Carlo simulations. When used from Brownian random walks to Levy flights, they are more elaborate (Yang and Deb, 2009a). Levy flights have been theoretically suggested to be the best search approach for revising targets, while intermittent search approach is best for non-revisiting targets (Bénichou et al., 2006, 2011).

Exploitation or intensification refers to the use of the established local search knowledge and solutions so that new search efforts will be channelled on the neighbourhood or local regions where the optimal solution can be located. However, this local optimum may certainly not be the global optimum. Exploitation strives to utilize strong local search information such as the mode shape (like convexity), the gradients, and the search history. A classic technique is the acclaimed Hill-climbing technique which intensively uses local gradients or derivatives.

2.3.2 Classification and Challenges of Metaheuristics

2.3.2.1 Classification of Metaheuristics

There are different ways to classify metaheuristics (as shown in Figure 2.4) based on the characteristics used to differentiate them (J. Shi and Zhang, 2018; Siddique and Adeli, 2015; Slowik and Kwasnicka, 2018; Talbi, 2009; Yang, 2014). They include:

- Based on searching behaviour: Metaheuristic can be classified into local or global metaheuristics based on their searching behaviour. Local search metaheuristics do not necessarily find the global optimum as the usually converge toward a local optimum. Such algorithms are usually deterministic and cannot escape local optima. A common example of local search algorithms is the Simple Hill Climbing algorithm. Contrarily, efforts are always toward finding a robust global best for any problem, although such global best is not always possible to be established. The local search algorithms are not suitable for global optimization; thus, global search algorithms are employed for global optimization. Most times, modern metaheuristics are intended for global optimization but they are not always efficient.

- Based on solution: Metaheuristics can be classified based on the number of solutions that interacted when trying to find the optimal solution. The single-based solution, also called trajectory methods, manipulate a single solution. Population-based methods iterate and manipulate a whole family of solutions. Single-based solution metaheuristics (e.g. Tabu search and Simulated annealing) are intensification oriented, while population-based (e.g. Particle Swarm Optimization, and Firefly Algorithm) focus more on search space exploration.

- Based on inspiration: Metaheuristics are also classified based on their original source of inspiration. Most metaheuristics are typically inspired by natural occurrences which are exploited by these algorithmic approaches in order to provide an efficient solution to optimization problems. Some of the nature-inspired algorithms (NIAs) are Particle Swarm Optimization algorithm, Genetic algorithm, Ant Colony optimization algorithm, and Simulated Annealing algorithm. The non-nature-inspired algorithms (NNIAs) include Tabu Search, Scatter Search, Iterated Local Search, and Grasp algorithms. These NNIAs are inspired by the consideration of the efficient solution they offer for optimization problems.

- Based on objective: Metaheuristics can be classified into single objective (SOM) and multi-objective metaheuristics (MOM) based on the number of conditions or objectives they tend to solve. The SOM mainly aims to find the best solution which represents the minimum or maximum value of a single objective function that converged all other objectives into one. These metaheuristics are useful tools that provide decision makers with information on the nature of the problem, but they usually do not offer a set of alternative solutions which trade different objectives against each other. Contrarily, MOM with conflicting objectives cannot offer a single optimal solution as these conflicting objectives interact to provide a set of compromised solutions. These solutions are largely referred to as trade-off, non-inferior, non-dominated, or Pareto-optimal solutions.

- Based on memory: Another attribute of metaheuristics is their ability to use previous search experiences (memory) to influence the future search direction. Memory-Less algorithms (e.g. simulated annealing) use only the information of the current state of search, while most of the other metaheuristics use some or all information gathered during the iterative process.

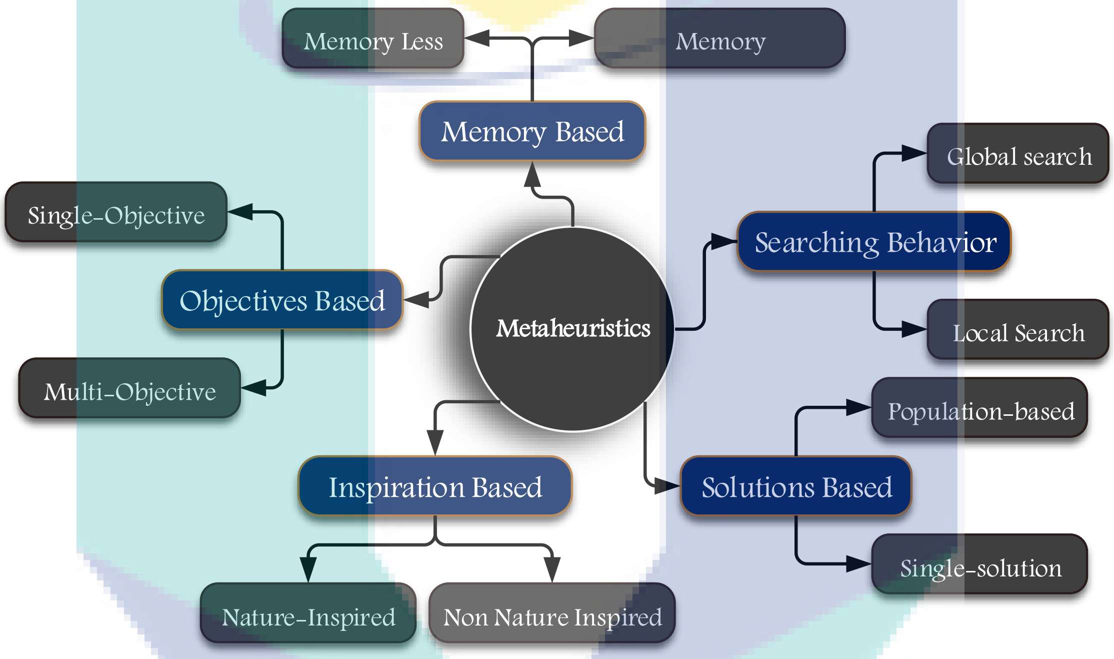


Figure 2.4 Classification of metaheuristics

2.3.2.2 Challenges of Metaheuristics

Metaheuristics are attracting much research interests, as manifested in the yearly emergence of new algorithms, and new techniques being explored (Fister et al., 2013; Yang, 2010, 2010b, 2011). However, there are several challenges facing the designing of metaheuristics, such as:

- **Premature Convergence**

Optimization algorithms which fail to reach new solutions or keep generating solutions from a relatively small subset of the solution space are said to have converged.

Global optimization algorithms must converge at some points with time, but a major problem of the global optimizers is that it is not often possible to determine the position of the current best solution (whether in the local or global optimum) and thus if the achieved convergence is acceptable. Hence, it is usually not clear whether to stop the optimization process, focus on fine-tuning the current optimum or examine other regions of the solution space. This is further complicated if there are multiple (local) optimal, i.e. if the problem is multimodal (Figure 2.5).

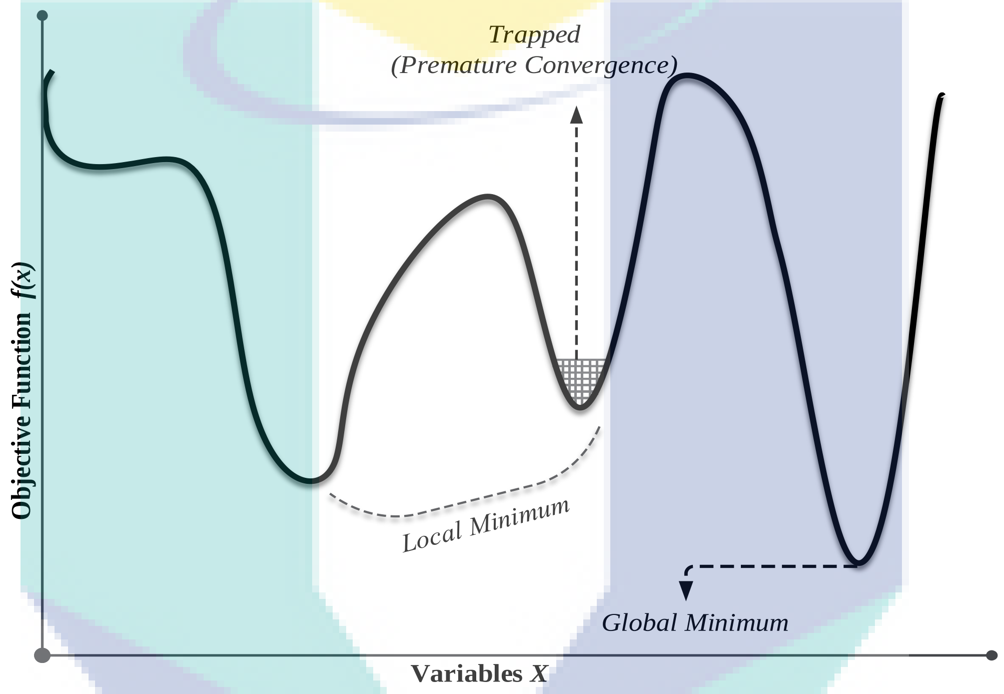


Figure 2.5 Premature convergence

Mathematical functions are said to be multimodal if they have multiple minima or maxima (Andrei, 2008; Jamil et al., 2013). However, objective functions f are said to be multimodal if they have multiple (local or global) optima. This depends on the definition of the term "optimum" with respect to the existing optimization problem. A metaheuristic is said to have prematurely converged to a local optimum if can no longer explore the other regions of the solution space, and there is no other region may yield a better solution to the current one (Ursem, 2003; Weise et al., 2009) . Figure 2.5 illustrates a premature convergence of metaheuristics.

- **Balancing Exploration and Exploitation**

As earlier mentioned, exploitation and exploration are two key components for any metaheuristic algorithm. Metaheuristics converge so quickly in the presence of too much intensification. This can lead to a local optimum or a wrong solution to a given problem. Furthermore, it can reduce the chances of establishing the true global best. Contrarily, too much diversification can increase the chance of finding the true global best, although it may slow the process with a significantly lower convergence rate. Therefore, a balance must be ensured between these two components (exploitation and exploration). Furthermore, it is not just enough to focus on only exploitation and exploration, there is also a need to select the best solutions using use a proper criterion. A common criterion used is the *survival of the fittest.* This implies a continuous updating of the current found best. Additionally, certain elitism is usually used to ensure the retention of the best solutions and its passage to the subsequent generations (Blum and Roli, 2003; Črepinšek et al., 2013; Holland, 1975; Yang, Deb, and He, 2013; Yang et al., 2017).

Effective algorithms usually have a mechanism of properly balancing exploitation and exploration. It should be noted that the Naıve 50:50 balance is not optimal (Yang, 2011), and more studies are needed in this area of metaheuristics.

- **Randomization Techniques**

To further analyze metaheuristics in detail, emphasis can be laid on the type of randomness employed by an algorithm. For instance, in a deterministic algorithm, the simplest and efficient method is the introduction of a random starting point. A good example is the well-known Hill-Climbing with a random restart. It is a simple and efficient strategy, and in most cases, easy to implement. Randomness can be efficiently introduced into an algorithm by incorporation into different algorithmic components. Various probability distribution such as uniform, Levy, or Gaussian distributions can be deployed for randomization (Agarwal and Mehta, 2014; Talbi, 2009; Yang, 2011). Randomization, in essence, is an important component of global search algorithms. Obviously, it still remains an open question that what is the best way to provide sufficient randomness without slowing down the convergence of an algorithm(Yang et al., 2017).

- **Tuning of Algorithm-Dependent Parameters**

The parameters of most metaheuristics are algorithm-dependent, and the performance of these algorithms depends on the appropriate setting of the values of these algorithm-dependent parameters. A challenging factor in setting these values has always been the selection of the right parameter values to be used in these algorithms, and how to tune these parameters to ensure a maximum algorithmic performance. In optimization studies, parameter tuning in itself is a tough task, but based on the existing literature, there are two major ways of parameters tuning. One approach is to run an algorithm with some trial values of the key parameters, and the aim of the test runs is to get good settings of these parameters. These parameters are then fixed for more extensive test runs involving a similar type of problem or larger problems.

The second approach is the use of an established and tested algorithm to tune the parameters of other new algorithms. Then, an important issue arises; if algorithm A (or tool A) was used to tune algorithm B, what tool or algorithm should be used to tune algorithm A? Let’s say algorithm C should be used to tune algorithm A, then, what tool should be used to tune algorithm C? In fact, these key issues are still currently being researched (Eiben and Smit, 2011; Yang et al., 2017; Yang, Deb, Loomes, et al., 2013), but it should be more appropriate to design parameter-free metaheuristics.

- **Necessity for Large-Scale and Real-World Applications**

Metaheuristic computations are very successful in solving many practical problems. However, the size of these problems in terms of number of design variables is relatively small or moderate. In the current literature, studies have focused on design problems with about a dozen of variables or at most about a hundred. It is rare to see studies with several hundreds of variables. In contrast, it is routine in linear programming to solve design problems with half a million to several millions of design variables. Therefore, it remains a huge challenge for SI-based algorithms to be applied to real-world large-scale problems. Accompanying this challenge is the methodology issue. Nobody is sure if the same methods that work well for small problems can be directly applied to large-scale problems. Apart from the differences in size, there may be other issues such as memory capacity, computational efficiency, and computing resources needing special care. If it is not possible to extend the existing methods to deal with large-scale problems effectively, often not, then what are the options? After all, real-world problems are typically nonlinear and are often large-scaled (Agarwal and Mehta, 2014; Caraffini et al., 2017; Yang et al., 2017). Further and detailed studies are highly needed in this area.

2.3.3 No Free Lunch Theorem

It is most likely that no optimization algorithm can be efficient and effective on all problems. There are specialized algorithms for specific problems; likewise, there are generalized methods which are outperformed by the specialized algorithms, but can achieve acceptable results in several kinds of problems. Wolpert and Macready have proposed the 'No Free Lunch Theorem (NFL)' (Wolpert and Macready, 1997) for search and optimization algorithms. For a given problem ϕ, the conditional probability for an algorithm alg to find global optima $gbest$ with iteration time itr is set as $P(gbest\,|\phi, itr, alg)$. The NFL proves that the sum of all conditional probabilities over all possible problems on finite domain is always identical for all optimization algorithms. The average performance over all given problems is independent of the applied algorithm. That is, for two optimizers alg_1 and alg_2:

$$\sum_{\forall\phi} P(gbest \mid \phi, itr, alg_1) = \sum_{\forall\phi} P(gbest \mid \phi, itr, alg_2) \qquad 2.6$$

For alg_1 to outperform alg_2 in one optimization problem, alg_1 must be inferior in another problem. Figure 2.6 shows a crude sketch of NFL theorem. It is impossible for any method to always outperform non-repeating random walks (Weise, 2009). Algorithms can only achieve good results in certain types of problems.

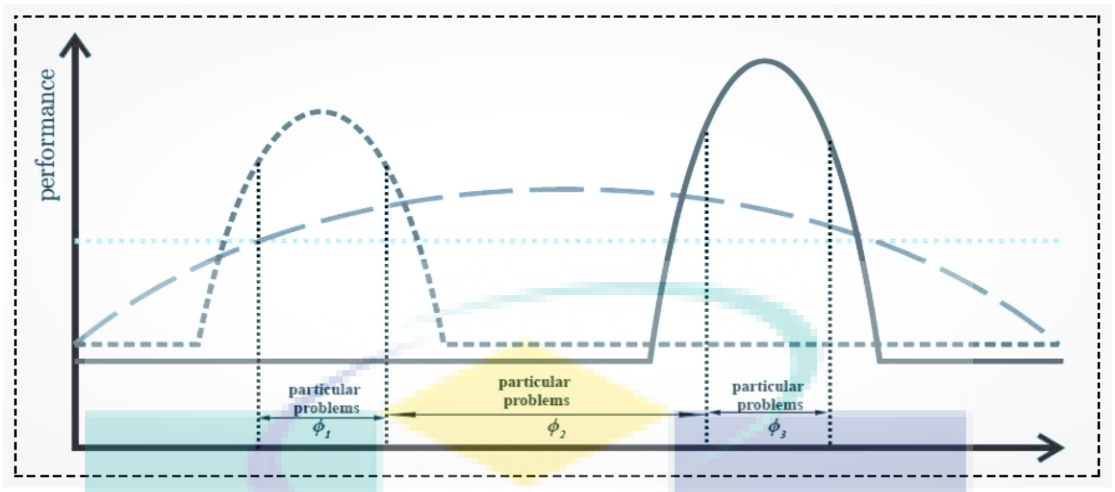


Figure 2.6 The 'No Free Lunch Theorem' (Engelbrecht, 2007)

2.4 Nature-Inspired Algorithms

The nature-inspired algorithms (NIAs) comprise of a branch of metaheuristics where most of its algorithms are nature-inspired. The maximum sources of inspiration have always been from chemistry, biology, or physics, but nature has been the main source of inspiration. Hence, most of the new algorithms are often called nature-inspired algorithms. Most of these nature-inspired frameworks are based on some observed behavior of biological systems. Consequently, a large portion of the nature-inspired algorithms are biologically or bio-inspired (Fister et al., 2013; Koziel and Yang, 2011).

A special class of swarm-intelligence-based algorithms has been developed among the bio-inspired algorithms; thus, some bio-inspired algorithms can now be referred to as swarm intelligence-based algorithms. Some of the swarm-intelligence-based algorithms are Cuckoo search, Bat algorithm, and Artificial Bee Colony (Yang and Deb, 2009b; Yang, 2010; Karaboga, 2005; Karaboga and Basturk, 2007). Some algorithms have been based on inspiration from physical and chemical systems, such as Simulated Annealing, Central Force Optimizer, and Gravitational Search algorithm (Kirkpatrick et al., 1983; Formato, 2014; Yadav et al., 2016). Some algorithms are even music-inspired, like Harmony search (Zong et al., 2001). The currently existing intelligent algorithms have been categorized into four: bio-inspired (but not SI-based), swarm intelligence (SI)-based, physics/chemistry-based, social evolution based, and others (Koziel and Yang, 2011).

2.4.1 Swarm Intelligence

Swarm Intelligence (SI) is an intelligent, innovative, distributed system for handling optimization problems. They are developed based on inspirations from biological systems such as flocking, herding or swarming behaviours. The conventional computing paradigms have shown difficulty in optimizing real-world problems. The real-world problems are often characterized by incomplete noisy data or multimodality as a result of their inflexible architecture. Natural systems have been explored over time to handle optimization problems. These natural systems usually contain numerous simple elements which works collectively to produce complex emergent behaviours. The natural computing frameworks can be utilized in situations where the traditional computing frameworks have failed. Swarm intelligence (SI) belongs to one of such natural computing paradigms.

Over the years, biologists have shown a great interest in studying the collective behaviour of social animals such as fishes, insects, birds, and mammals. The French biologist Pierre-Paul Grasse provided the first ever theoretical explanation of the collective behavior of social animals. In 1984, Grasse reported the collective behavior of African termites. The first flocking model was developed in 1987 by Craig Reynolds as a bio-inspired computational model for the simulation of the animation of a flock of entities called Boid (Reynolds, 1987). Collective patterns and decision making was presented by Deneubourg and Goss (1989). In 1991, the food foraging of ants and the shortest route to their food sources were studied by Deneubourg et al. (1991). Swarm intelligence was first introduced by Beni and Wang (1991) in the field of cellular robotic systems. In 1992, Marco Dorigo introduced the Ant Colony Optimization (ACO) algorithm as a heuristic that was inspired by the behaviour of ant when searching for food (Dorigo, 1992). Later, the PSO was developed based on the social flocking behaviour of birds by Eberhart and Kennedy (1995).

Swarm Intelligence (SI) is a growing research field in the natural computing paradigm. It deals with systems (natural and artificial) made up of many simple individuals. Each of the components is coordinated using self-organization and decentralized control. SI is the outcome of collective behaviours of simple individuals that interact with each other and with their environment. They can solve complex, discontinuous, multimodal, non-differentiable, and distributed optimization problems. It serves as an alternative approach towards designing systems that are either impossible or near impossible by conventional optimization algorithms. A typical SI algorithm has the following properties:

- They are made up of several simple relatively homogeneous individuals.
- These individuals interact based on simple behavioural principles.
- These individuals directly exchange information among themselves or via the environment.
- The overall system behaviour results from the individual interaction of these individuals with each other and with their immediate environment.
- Individuals have the division of labour and distributed task allocation system among them.
- The individuals act in a coordinated manner in the absence of a coordinator or an external controller.
- Each individual has a stochastic behaviour that depends on its local perception of the neighbourhood.

Because of these properties, SI has maintained a prominent presence in computational research. The generalized pseudocode of swarm intelligence-based metaheuristics is depicted in Figure 2.7.

```
Swarm Intelligence Based Algorithm
1.    Define Objective Function $f(x)$
2.    Initialize the structural and controlling parameters
3.    Initialize the positions for all individuals randomly
4.    Evaluate the fitness values for all individuals
5.    Repeat
6.        Update Controlling parameters
7.        Update the positions of the individuals
8.        Rank the individuals
9.        Get the global best solution "$gbest$"
10.   While $StopCondition$ is not satisfied
```

Figure 2.7 The generalized pseudocode of SI-based algorithms

2.4.2 State of the Art in NIAs and SI

Evidently, several attempts have been made towards the designing of metaheuristics. In the last two decades precisely, several swarm intelligence algorithms were designed, and most of these intelligence algorithms focused on optimization.

The bio-inspired algorithms are algorithms which were designed based on the intelligent behaviour of groups of animals, or insects. Many algorithms currently available in the literature are in this category. Some of the bio-inspired algorithms are the Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) algorithms. Efforts are still currently being made towards the development of more animal-inspired algorithms. In this section, a review of the most common animal-inspired metaheuristics (ACO, PSO, Artificial Bee Colony (ABC), Firefly Algorithm (FA), Bacterial Foraging Optimization (BFO), Cuckoo Search Algorithm (CSA), Bat Algorithm (BA), Fish Swarm Optimization (FSO), African Buffalo Algorithm (ABO), and Grey Wolf Optimizer (GWO) will be carried out.

In recent times, ACO is a common technique for solving combinatorial optimization problems. ACO was introduced by M. Dorigo in the early 1990s (Colorni et al., 1991; Dorigo et al., 1996; Socha and Dorigo, 2008) as a search technique inspired by the swarm intelligence of ant colonies using pheromone as a chemical messenger. When searching for food, ants initially explore the area surrounding their nest in a random manner. While exploring, the moving ants leave a pheromone trail on the ground. This pheromone evaporates with time later serve as a guide for the ants probably based on the pheromone concentrations on the explored paths. Such an indirect communication strategy enables the ants to explore the shortest route to the food source or to their nest.

PSO was developed by Eberhart and Kennedy as a is a swarm intelligence technique which was based on the social behaviour of birds (flocking) and fishes (schooling) (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995). For every swarm, movements are made with variable velocities to locations with better experiences or food compared to the previous explored locations. In the PSO, there is no explicit selection function and this is compensated using leaders as a guide during search. The location of every particle is considered as a possible solution in the search space, and a solution update is often accomplished by updating the position of a particle. PSO uses real-number randomness and global communication among the swarm particles rather than using crossover or mutation. PSO has been proven as an effective optimization algorithm, especially in a large search space. Generally, the updating of a particle requires the updating of two properties, which are the velocity and the location of the particle.

The Artificial Bee Colony (ABC) which was developed by Karaboga (2005) and Dervis Karaboga & Basturk (2007), was inspired by the natural behaviour of bees in their colonies. In the bee colony, there are three bee groups and bees belonging to each group play a different role. The employed (forager) bees are responsible for the current food source, they have all the information (such as the direction and distance from the nest, the profitability) regarding the particular food source and share the information based on the profitability of the food source. The onlooker (observer) bees utilize the information shared by the employed foragers to establish a food source in the nest. The scout bees are responsible for discovering new sources of food to be exploited. Information regarding the quality of food sources is shared among the bees in the dancing area; food sources with more profitability receive more information, and based on this information, the onlookers can decide on the next source of food to be explored. Randomization is carried out by the scout and employed bees mainly by mutation.

The BFO approach is of the view that animals forage for nutrients in a way that the time spent on foraging is equivalent to the energy gained from the task per unit time. Foraging strategies comprise of the approaches towards food sourcing, handling, and ingestion. Foraging animals act based on the limitations presented either by its own physiology (such as sensing and cognitive capabilities) or the environment (such as the number of prey, the risk from predators, and the physical characteristics of the search space). This idea led to the development of this approach as an optimization method (Passino, 2002).

FSO is an optimization approach based on the swarm behaviour of fishes when searching for food. The following fish behaviours can be considered when searching for food: random behaviour – this is when fishes search randomly for either food or a companion; searching behaviour – this is when fishes discover regions with more foods; swarming behaviour – this is when fishes naturally swarm to avert danger; chasing behaviour – this is when fishes dangle to the location of a new source of food; leaping behaviour – this is when fishes are stagnated in a region and there is a need to search for food in other regions (Li et al., 2002).

The FA is an optimization algorithm that is based on the behavior and light flashing patterns of tropical fireflies. It is an easy, simple, and flexible algorithm to use. Ideally, it operates based on the rules that i) there is no sex variation among the fireflies (unisex), firefly are attracted to each other irrespective of their sex; ii) the brightness of the emitted light determines the attractiveness of a firefly (both parameters decrease as the distance between the fireflies increase, and in the absence of any brighter firefly, the swarm will be moving randomly); iii) the brightness of the light emitted by a firefly is a function of the objective function (Yang, 2010; Yang, 2009).

BA was proposed by Yang in 2010 as a swarm-based metaheuristic algorithm which was inspired by echolocation, a type of sonar that guides the flying and hunting behavior of bats. Bats can move effortlessly in complete darkness, and can distinguish different types of insects in complete darkness (Yang, 2010a). The rules of BA are: All bats sense their location and distance via echolocation and the echolocation of a bat is considered as a possible solution to an optimization problem. Bats randomly fly at velocities with different frequencies based on their position (from a minimum f_{min} to a maximum frequencyf_{max}) or based on their wavelength λ and loudness A to search for prey. They bats can automatically adjust their rate of pulse emission r or the wavelength (or frequency) of their emitted pulses based on their proximity to their target. Loudness can vary from a large positive value A_0 to a minimum constant valuef_{min}.

CS is a recently proposed metaheuristic algorithm (Yang & Deb, 2009) which was inspired by the reproduction behavior of cuckoos. Cuckoos basically lay their eggs in the nests of other birds (may or may not be the same species). These eggs are either destroyed by the host bird or abandoned (together with the nest) when it realized it is not its own eggs. This behavior resulted in cuckoos laying eggs that resemble those of the host birds (Fister et al., 2014). In an effort to apply this behavior as an optimization tool, three rules were considered by the authors: i) each cuckoo can lay only one egg which is dumped in a random nest; this egg represents a set of solution; ii) a portion of the nests that contains the best eggs (best solutions) will be represented in the next generation; iii) there is a fixed number of nests, and there is a chance of the host bird discovering an alien egg and if this happens, the egg can either be discarded or the whole nest will be abandoned, resulting in the building of a new nest in a new location.

ABO algorithm is a simulation of the movement and herd management structure of the African buffalos as they move around the African deserts, savannah, and forests in search of lush grazing lands(Azrag et al., 2017; Odili et al., 2015). The African buffalos manage their large herds of, sometimes, up to 1200 using two major sounds: /waaa/ asking the buffalos to further explore the landscape and /maaa/ sounds that require the buffalos to further exploit their immediate environments for possible solutions.

The Grey Wolf Optimizer (GWO) is a swarm-intelligence algorithm inspired from the hierarchical leadership and natural hunting behavior of Grey wolves (Mirjalili et al., 2014). The Grey wolves are regarded as apex predators with an average size of 5–12 wolves per group. In the GWO hierarchy, the most dominant member of a group is designated $alpha$ while the subordinates are designated $beta$ and $delta$. This helps to control the rest of the wolves designated $omega$ in the hierarchy. The mathematical representation of the grey wolves hunting mechanism consists of tracking, chasing, and approaching a prey; pursuing, encircling, and harassing the prey to a standstill; and then, attacking the prey.

These are the recent and upcoming class of optimization algorithms which are based on the concept of simulating and mimicking social human learning behaviours (or social evolution). The concept of social algorithms was first introduced by Reynolds & Sverdlik, (1994) based on the principle that the evolution of individuals through cultural evolution is faster than biological or genetic evolution alone. Humans can quickly improve their intelligence by adapting to mannerisms and behaviours via observing or imitating others. The human ability to cooperate and co-exist as a group adds to their collective intelligence. This has been the basis for most studies on the formalization of the recent socio-inspired algorithms such as Society and Civilization Optimization algorithm (SCO), League Championship algorithm (LCA), Social Emotional Optimization algorithm (SEOA), Anarchic Society Optimization algorithm (ASO), Imperialist Competitive algorithm (ICA), Teaching–learning-based optimization (TLBO), Cultural Evolution algorithm (CEA), Election Campaign Optimization algorithm (ECO), Cohort Intelligence (CI), Election Algorithm, Social learning optimization (SLO), Social Group Optimization (SGO),), Soccer League Competition

Algorithm (SLC), Ideology Algorithm (IA), and Socio evolutional & learning optimization algorithm (SELO).

The human ability to mutually interact is fundamental and prevalent in all human and insect societies. Humans can adapt and improve faster through social interactions than through biological evolution solely based on genetic inheritance. This is the basis for the optimization algorithm proposed by Ray & Liew (2003) which utilizes the intra and intersociety interactions within a society and the civilization model to solve single objective optimization problems. In this algorithm, a society is represented by a collection of points in the parametric space, while civilization is represented by a set of all such societies at any time.

The ICA was first proposed by Atashpaz-Gargari & Lucas (2007) for the simulation of the socio-political behaviours witnessed across imperialist nations which compete to assert dominance over weaker colonies or empires. This imperialist competition finally results in improving the strength and power of the imperialists whilst gradually collapsing the weaker empires, and finally leading to a state of convergence. The LCA which was established by Husseinzadeh Kashan (2014) and Kashan (2009) was formulated based on the competition between teams competing in league matches. Artificial teams (representing solutions) weekly compete (representing iterations) based on a league schedule. A stronger team (probably with a higher fitness value) gradually wins the competition at the end of the playing season (stopping condition).

SEOA is a swarm based socio-inspired metaheuristic which simulates a virtual being who based on his emotional conviction, wishes to achieve a higher societal status (Xu et al., 2010). His current behaviour is determined by an emotional index while the society determines the status of this current behaviour (better or worse), thus, affecting the value of this emotion index (parameter). Lv et al. (2010) introduced ECO algorithm which was inspired by the mannerisms of political candidates during an election campaign. They inspire the voters to vote for candidates with a better prestige (better function value). At the end, a stronger candidate receives the maximum number of votes. The ASO algorithm is another optimization algorithm which was inspired by the human greedy and disorderly behaviour when trying to achieve their goals (Ahmadi-Javid, 2011). People can behave disorderly just to find better solutions in the solution space.

TLBO is an optimization method that is inspired by the teacher's influence on the learning outcome of their students. There are two aspects of the methodology - the teacher's phase and the learner's phase. The learners gain more knowledge (improve solution quality) by tapping from the teacher's knowledge or through interacting with their peers (Rao et al., 2011; Rao, Savsani, & Vakharia, 2012). Kuo and Lin proposed a framework for Cultural Evolution Algorithm (CEA) in 2013. They stated that a species can learn or evolve either through group consensus, innovative learning, self-improvement, or individual learning. The authors have stated the mathematical model for these learning modes. (Kuo and Lin, 2013).

SLC is inspired from the competitive behaviour among teams and players in soccer league matches and has been effectively used to solve discrete and continuous optimization problems (Moosavian and Kasaee Roodsari, 2014). The Election Algorithm is another algorithm which mimics an election scenario, comprising of electoral parties, the voters and the candidates (Emami and Derakhshan, 2015).

CI is a socio-inspired metaheuristic which mimics the self-learning behaviour shown by candidates in a group (the candidates cooperate and compete with each other to achieve individual goals) (Kulkarni et al., 2016). SLO is an optimization approach based on the different levels of human evolvement (genetic evolution and cultural evolution). Cultural evolution is believed to influence genetic evolution in future generations, and this influence helps to accelerate human intelligence. SGO is a recent algorithm which was inspired by the human behaviour when trying to collectively solve a complex task (Satapathy and Naik, 2016). This mannerism of being influenced by better behaviours and person, coupled with necessary modifications, helps to address complex problems.

Another innovative algorithm IA (Huan et al., 2017) is inspired from the idea how certain beliefs become the guide for individuals in a society to achieve their goals. IA elicits this idea through a political scenario where individuals follow their political ideologies and compete with members of their political party as well as with leaders of other political parties in their will to excel. Recently, SELO algorithm has been proposed, inspired by the social learning behaviour of humans organized as families in a societal setup. It is the social tendency of humans to adapt to mannerisms and behaviours of other individuals through observation. SELO mimics the socio-evolution and learning of parents and children constituting a family. Individuals organized as family groups (parents and children) interact with one another and other distinct families to attain some individual goals (Kumar et al., 2017).

This thesis focuses on only five well-known nature inspired metaheuristics, they are PSO, ABC, FPA, GWO, and FA. These metaheuristics are selected based on their main characteristics such as randomization techniques, the source of inspiration, the type of the algorithm, and the structure of the algorithm as summarized in Table 2.1. Meanwhile, Table 2.2 summarizes them in terms of exploration, exploitation, and their drawbacks. These algorithms will be used for comparison in chapter four.

Table 2.1 Metaheuristics and their characteristics

Alg.	Randomization	Source of Inspiration	Structure	Global Best	Evolution
PSO	Uniform Distribution	Based on the social behaviour of birds (flocking) and fishes	• The particles attracted to each other based on the velocity equation. • Each particle represent a local best solution, while there is only one global best solution.	g^*	Swarming towards g^*
ABC	Uniform Distribution	It is inspired by the behavior of honey bees when seeking a quality food source	• The swarm consists of three types of bees, employee, onlooker, and scout. • Employee represent the local search, while the scout represent the global search.	g^*	Information exchange during the process of honey collection
FA	Gaussian Distribution	It is based on the behaviour and light flashing patterns of tropical fireflies	• The fireflies are attracted towards the brightest firefly based on its intensity. • It is an intrinsic multiple swarm system because the initial swarm can automatically subdivide into multiple swarms, due to the fact that local attraction is stronger than long distance attractions.	Brightest	Attraction
GWO	Uniform Distribution	It simulates the leadership hierarchy and hunting mechanism of gray wolves in nature.	• It follows the pack hierarchy for organizing the different roles in the wolves pack. • The swarm is divided into four groups (Alpha, Beta, Delta, and Omega).	Alpha (α)	Leadership grading and hunting
FPA	Uniform Distribution & Levy Flight	Flower pollination process is associated with the transfer of pollen by using pollinators such as insects, birds, bats,	• It is a population based algorithm, produces the optimal reproduction of plants by surviving the fittest flowers in the flowering plants. • Based on reproduction process via pollination.	g^*	Constancy and Similarity

Table 2.2 Exploration, exploitation, and drawbacks of five metaheuristics

Alg.	Exploration	Exploitation	Drawbacks	References
PSO	Velocity update of particles.	Update the particle positions towards the global best particle.	• It can be easily trapped in the local optimum in high-dimensional space. • Its convergence rate is low during iterations. • It has three controlling parameters, c_1, c_2, w.	(El-Shorbagy and Hassanien, 2018; Mirjalili and Hashim, 2010; Mirjalili et al., 2012; Y. Zhang and Wu, 2011)
ABC	Random search of scout.	Neighbourhood search is performed by employed and onlooker bees.	• It converges slowly in the process of searching and easily suffers from premature. • The value of $limit$ affect the exploration process.	(Hala M. Alshamlan et al., 2015; Hala Mohammed Alshamlan, 2018; Karaboga et al., 2014)
FA	Random move of the best firefly.	Firefly movement according to attractiveness.	• It has two controlling parameters, a and γ need to be tuned. • Computational time is high due to too many attractions.	(Gandomi et al., 2013; H. Wang et al., 2017; Yaseen et al., 2017; L. Zhang et al., 2017)
GWO	Tracking the prey, also the value of C influences on the exploration which represents the weight of the prey in defining the distance.	The Grey wolves conclude the hunt by attacking the prey when it ceases to move. It allows the position of its search agents to be updated based on the location of the alpha, beta, and delta; and attack towards the prey.	• It faces the problem of premature convergence due to the problem of stagnation of wolf pack in local optima. • Low capability to handle the difficulties of a multimodal search landscape, as it seems that all α, β, and δ tend to converge to the same solution.	(Gupta and Deep, 2018; Heidari and Pahlavani, 2017)
FPA	The ability of the pollinators to travel long distances make it possible for the algorithm escape local optimum and subsequently explore a wider solution space.	Flower consistency ensures the selection of the same flower species and thus, ensures a speedy convergence.	• It consists of three controlling parameters, $p, \gamma, and\ \lambda$. • It has the problem of slow convergence, low precision and easy to fall into a local optimum.	(Benkercha et al., 2017; Chakraborty et al., 2015; Salgotra and Singh, 2017; X. S. Yang, Deb, and He, 2013)

2.5 Gap Analysis

Optimization is a process of minimizing or maximizing the objective function of an optimization problem by finding the best values for its variables. Optimization is applicable in every aspect of life, ranging from engineering, computer science, finance applications to decision making. Optimization is paramount in almost all engineering and industrial application, from minimizing energy cost and consumption to maximizing output, profit, efficiency, and performance (Aydilek, 2018; Koziel and Yang, 2011; Kumar et al., 2017).

Most optimization engineering and industrial problems, under stringent constraint, are highly nonlinear and thus, often NP-hard. It is difficult to find optimal solutions to such problems, if not impossible. Optimization problems are problems that require the determination of a set of unknown variables $\{x\}$ in order to satisfy the number of constraint functions and to minimize the objective function $f(x)$. Optimization processes aim to establish the values of these variables to ensure the maximization or minimizing of the objective function (Luke, 2013; Saka et al., 2013).

A complete search algorithm searches the solution space for all the possible values to be assigned to the variables, but their major drawback that they require much time. on the other hand, incomplete search methods search the solution space in either a systematic or non-systematic manner (Barták et al., 2010). Metaheuristic techniques have over the years become popular due to several advantages such as their flexibility, local optima avoidance, and gradient-free mechanism. These advantages rely on the ability of metaheuristics to consider and handle optimization problems by only considering the inputs and outputs. This implies that metaheuristics consider an optimization problem as a black box. Hence, it is not necessary to calculate search space derivative, and this accounts for their suitability for solving a range of optimization problems.

Metaheuristics benefit from random operators since they belong to the class of stochastic optimization techniques. This enhances their ability to escape local solutions when handling real problems that may usually contain relatively large local optima. These advantages have made metaheuristics suitable for application in different science and industrial fields (Mirjalili et al., 2017; Siddique and Adeli, 2015; Slowik and Kwasnicka, 2018).

It is evident from the literature that several popular and efficient optimization algorithms have been developed and will keep being developed. However, concurring to the ‘no free lunch optimization theorem’ (Köppen et al., 2001; Wolpert and Macready, 1997), no single optimization algorithm can suite all optimization problems, meaning that there is no universal algorithm/method that can be perfectly used to solve all optimization problems. A class of algorithms will perform better than others when applied to defined problems with specific objective functions.

A major problem now is how to identify specialized optimization algorithms which will offer better performances in specialized scenarios. Thus, it can be said that the development of new optimization algorithms will always be paramount as it will provide a ground for the development of newer prospective algorithms which could suite specific classes of optimization problems. These new algorithms may also perform better than most of the existing algorithms when solving specific optimization problems (Ho and Pepyne, 2002; Slowik and Kwasnicka, 2018).

Several metaheuristic-related issues have been addressed in this chapter. First, all metaheuristics have two main components (exploration and exploitation) which exerts an effect on the search space. Finding the optimal balancing of these components especially involving large scale multidimensional problem is still a difficult endeavour. Second, most metaheuristics contain one or more control parameters that enhance their ability to search for best solutions. However, the optimal values for these parameters are difficult to be determined; hence, there is a need to design new parameters-free metaheuristics (Slowik and Kwasnicka, 2018; Yang et al., 2016).

In this study, two nature-inspired solutions are proposed to overcome the above-mentioned problems. The first solution is a new multi-swarm cooperative scheme for balancing the exploration and exploitation of metaheuristics called “Meeting Room Approach (MRA)”. This is inspired by the human interaction when finding solutions to life problems. The second solution is a parameter-free nature-inspired metaheuristic called “nomadic people optimizer (NPO)”. This optimizer simulates the life of nomads when searching for food sources in the desert. The main rules of the proposed algorithms are inspired from the life of Bedouins. The NPO consists of several clans or swarms which communicates via the proposed MRA. Both solutions will be further explained in the next chapter.

2.6 Summary

The success or failure of metaheuristic algorithms depends on its ability to establish good balancing between exploration and exploitation. A poor balance between exploration and exploitation may result in weak optimization, which may cause the algorithm to be trapped in local optima, stagnant or leap over optimal solution. The metaheuristics which have been developed in the literature contain one or more controlling parameters to enhance the balancing between these two main components. Finding the optimal values for these parameters is nearly impossible, therefore, a parameter-free metaheuristic will be better in terms of complexity, and can be used in different domains without any additional adaptive methods.

The number of decision variables has a great effect on the searching algorithms. The problems with only one decision variable are called univariate optimization problems, while the problems with more than one decision variable are called multivariate problems. Large-scale optimization problems are multivariate problems with a large number of decision variables, they main contain hundreds or thousands of variables. Most studies in the literature have focused on optimization problems with about 50 or less number of variables, and when compared to real-world problems, the observed dimensionality is significantly low. Meanwhile, the existence and scalability of any algorithm that can be directly used in large-scale problems are yet to be verified. It is recommended to test optimization problems with about 50 to thousands of variables.

This chapter presented the theoretical background of optimization in general, and metaheuristics in particular. The chapter also reviewed nature-inspired metaheuristics and most of the recent optimization algorithms. Finally, the issues in the field of optimization were analysed. In the next chapter, the proposed metaheuristic will be detailed, starting with its inspiration to the provision of its mathematical model.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

One of the hot existing research fields in computer science is the developing of algorithms or metaheuristics for solving the optimization problems. Nature has been widely been used as a source for developing optimization algorithms, these type of algorithms are called “Nature-Inspired Algorithms (NIAs)”. A mammoth number of NAIs have been proposed in the literature, most of them are inspired from biological systems such as flocking, herding or swarming behaviours. The conventional computing paradigms have shown difficulty in optimizing real-world problems.

The real-world problems are often characterized by incomplete noisy data or multimodality as a result of their inflexible architecture. Natural systems have been explored over time to handle optimization problems. These natural systems usually contain numerous simple elements which works collectively to produce complex emergent behaviours. The natural computing frameworks can be utilized in situations where the traditional computing frameworks have failed. Swarm intelligence (SI) belongs to one of such natural computing paradigms.

This remainder of this chapter describes the research methodology used to develop the proposed algorithm in solving normal and large-scale optimization problems. Section 3.2 offers an elaborated detail of research methodology adopted. Section 3.3 illustrates the benchmarks for testing and validation of the proposed algorithm. Then, section 3.4summarizes and concludes the chapter.

3.2 Research Methodology

The activities carried out in this study are grouped into three stages, which are the background and literature review stage, methodology stage, and benchmarking and analysis stage. Figure 3.1 shows these stages.

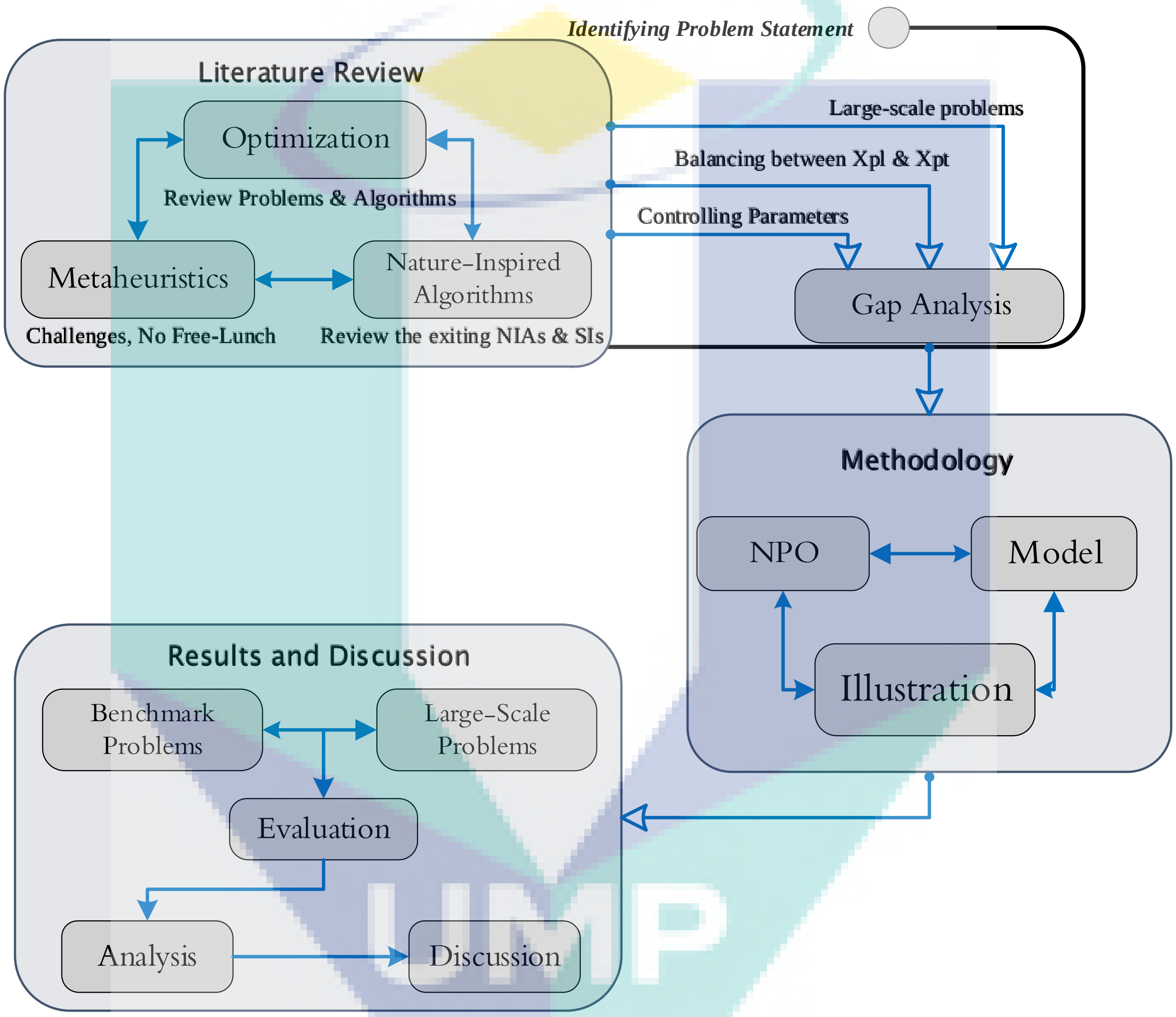


Figure 3.1 Research processes

3.2.1 Literature Review

In this stage, the current effort in the field of optimization was analyzed. The literature survey specifically focused on understanding the basics of metaheuristics and simultaneously identifying the strengths and drawbacks of the existing work in order to establish a focused research problem. This stages ends with identifying the gap, and determining the main issues faced by recent metaheuristics.

3.2.2 Methodology

This study mainly focuses on the development of an efficient swarm-based metaheuristic for solving global optimization problems. The proposed metaheuristic is a new nature-inspired algorithm that mimics the migration of nomads when searching for foods and water sources in the desert. In addition, it uses a novel multi-swarm approach for balancing its exploration and exploitation abilities. This stage involves the inspiration, modeling, design, and implementation of the new algorithm “Nomadic People Optimizer (NPO)” metaheuristic. The algorithm consists of five main steps; it is started by the initiation step and ended with the “periodical meeting” (to be detailed in the concept of the proposed multi-swarm approach).

3.2.3 Results and Discussion

The developed NPO was evaluated at this stage in two sets of experiments. In the first set of experiments, the overall performance of the NPO and the benchmarking algorithms was evaluated over a fixed number of iterations. Having completed a given number of iterations, the algorithms were also evaluated based on the achieved best fitness and mean values for each benchmarked function. The number of iterations employed in this study was fixed at 1000. In the second set of experiments, the algorithms were evaluated in terms of their convergence behaviour. In this case, the algorithms were run on various numbers of iterations and the achieved mean fitness values were established. Hence, their convergence behaviour was obtained based on the number of iterations. The proposed NPO algorithm was applied on a new combination of benchmarked functions, while its performance was compared to those of five renowned algorithms (PSO2011, ABC, GWO, FPA, and FFA).

3.3 Benchmark Functions

It is mandatory that the performance of any newly developed algorithm should be benchmarked and validated against that of other existing algorithms using a good set of test functions. Most researchers prefer to test the performance of their algorithms on a large test set, especially when optimization functions are involved. However, the effectiveness of one algorithm over others cannot solely depend on its ability to solve problems that are either too specialized or without diverse features. The evaluation of an algorithm demands the identification of the kind of problems that it had a better performance compared to others. This will help in determining the type of problems that the algorithm can be used to solve. This can only be achieved by using a test suite that is large enough to embrace a range of problems such as unimodal, multimodal, separable, non-separable, and multi-dimensional problems (Jamil et al., 2013; Qu et al., 2016).

This present study focuses on the test function benchmarks and their diverse features such as modality and separability. A function is multimodal if it has more than one local optimum, and are used to test the ability of an algorithm to escape being trapped in any local minima. If an algorithm is built with a poorly constructed exploration process, it cannot effectively search the function landscape, and this could result in having the algorithm stuck at local minima. The most difficult class of problems for most algorithms is the multi-modal functions with many local minima. The difficulty of different benchmark functions is expressed in terms of their reparability. Because each variable of a function in separable functions is independent of the other variables, they are generally easily solved compared to their inseparable counterpart.

To evaluate the performance of the NPO, 60 test functions were carefully selected in this study from several references (Andrei, 2008; Jamil et al., 2013). These test functions were divided into four groups (Unimodal Non-Separable (U-N) with 14 tests, Unimodal Separable (U-S) with 11 tests, Multimodal Non-Separable (M-N) with 26 tests, and Multimodal Separable (M-S) with 9 tests). We strongly recommend using this set of test functions by future researchers. Table 3.1 shows the general information about these tests, while the functions are given in Appendix A.

Table 3.1 Benchmark Test Functions used for evaluation

f_n	Name	Type	D	LB-UB	Optimal	f_n	Name	Type	D	UB-LB	Optimal
1	Ackley No.2	U-N	2	-32, 32	-200	**31**	Cross-In-Tray	M-N	2	-10 , 10	-2.0626
2	Beale	U-N	2	-4.5,4.5	0	**32**	Drop wave	M-N	2	-5.12,5.12	-1
3	Brown	U-N	30	-1 , 4	0	**33**	Egg Holder	M-N	2	-512,512	-959.64
4	Easom	U-N	2	-10,10	-1	**34**	Griewank	M-N	30	-600,600	0
5	Leon	U-N	2	-1.2,1.2	0	**35**	GoldStein-Price	M-N	2	-2 , 2	3
6	Matyas	U-N	2	-10, 10	0	**36**	Hartman 3	M-N	3	0 , 1	-3.86278
7	Powell	U-N	24	-4,5	0	**37**	Hartman 6	M-N	6	0 , 1	-3.3223
8	Schaffer No.1	U-N	2	-100,100	0	**38**	Holder-Table	M-N	2	-10 , 10	-19.2085
9	Schaffer No.2	U-N	2	-100,100	0	**39**	Keane	M-N	2	-10 , 10	-0.67368
10	Schaffer No.3	U-N	2	-100,100	0.001567	**40**	Levy No.13	M-N	2	-10,10	0
11	Schaffer No.4	U-N	2	-100,100	0.29258	**41**	Penalized	M-N	30	-50 , 50	0
12	Trid 6	U-N	6	-36,36	-50	**42**	Penalized No.2	M-N	30	-50 , 50	0
13	Trid 10	U-N	10	-100,100	-210	**43**	Perm	M-N	4	-4 , 4	0
14	Zakharov	U-N	30	-5 , 10	0	**44**	Powersum	M-N	4	0 , 4	0
15	Powell Sum	U-S	30	-1 , 1	0	**45**	Price No.1	M-N	2	-500,500	0
16	Quartic	U-S	30	-1.28,1.28	0	**46**	Price No.2	M-N	2	-10 , 10	0.9
17	Schwefel 2.20	U-S	30	-100,100	0	**47**	Price No.3	M-N	2	-500 , 500	0
18	Schwefel2.21	U-S	30	-100,100	0	**48**	Price No.4	M-N	2	-500 , 500	0
19	Schwefel2.22	U-S	30	-100,100	0	**49**	Shubert	M-N	2	-10 , 10	-186.7309
20	Schwefel2.23	U-S	30	-10 , 10	0	**50**	Shubert No.3	M-N	2	-10 , 10	-29.6733
21	Sphere	U-S	30	-100,100	0	**51**	Shubert No.4	M-N	2	-10 , 10	-25.7408
22	Step	U-S	30	-100,100	0	**52**	Alpine No.1	M-S	30	-10 , 10	0
23	Step No.2	U-S	30	-100,100	0	**53**	BohachevskyN.1	M-S	2	-100,100	0
24	Stepint	U-S	5	-5.12,5.12	0	**54**	Booth	M-S	2	-10 , 10	0
25	Sumsquares	U-S	30	-10 , 10	0	**55**	Branin	M-S	2	-5 , 5	0.39789
26	Ackley	M-N	30	-32 , 32	0	**56**	Egg Crate	M-S	2	-5 , 5	0
27	Bird	M-N	2	$-2\pi, 2\pi$	-106.7645	**57**	Michalewicz 2	M-S	2	$0 , \pi$	-1.8013
28	BohachevskyNo.2	M-N	2	-10 , 10	0	**58**	Michalewicz 5	M-S	5	$0 , \pi$	-4.6876
29	BohachevskyNo.3	M-N	2	-100,100	0	**59**	Michalewicz 10	M-S	10	$0 , \pi$	-9.6601
30	Camel Six-Hump	M-N	2	-5 , 5	-1.0316	**60**	Rastrigin	M-S	30	-5.12,5.12	0

Table 3.1 above presented the main information for the 60 benchmarking test functions used in this thesis. The equations of these test functions are presented in Table 3.2 below.

Table 3.2 The Equations for each benchmark test functions

f_n	Test	Objective Function
1	Ackley N0.2	$f(x) = -20e^{-0.02\sqrt{D^{-1}\sum_{i=1}^{D} x_i^2}} - e^{D^{-1}\sum_{i=1}^{D}\cos(2\pi x_i)}$ +20+e
2	Beale	$f(x) = (1.5 - x_1 + x_1x_2)^2$+$(2.25 - x_1 + x_1x_2)^2$+$(2.625 - x_1 + x_1x_2)^2$
3	Brown	$f(x) = \sum_{i=1}^{n-1}(x_i^2)^{((x_{i+1}^2+1))} + (x_{i+1}^2)^{((x_{i+1}^2+1))}$
4	Easom	$f(x) = -\cos(x_1)\cos(x_2)\exp[-(x_1-\pi)^2 - (x_2-\pi)^2]$
5	Leon	$f(x) = 100(x_2 - x_1^2)^2 + (1+1)^2$
6	Matyas	$f(x) = 0.26(x_1^2 + x_2^2)$-$0.48x_1x_2$
7	Powell	$f(x) = \sum_{i=1}^{D/4}(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} + 10x_{4i})^2 + (x_{4i-2} + 10x_{4i-1})^4 + 10(x_{4i-3} + 10x_{4i})^4$
8	Schaffer No1	$f(x) = 0.5 + \frac{sin^2(x_1^2 + x_2^2)^2 - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$
9	Schaffer No2	$f(x) = 0.5 + \frac{sin^2(x_1^2 - x_2^2)^2 - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$
10	Schaffer No3	$f(x) = 0.5 + \frac{sin^2(\cos(\lvert x_1^2 + x_2^2\rvert))^2 - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$
11	Schaffer No4	$f(x) = 0.5 + \frac{cos^2(\sin(\lvert x_1^2 + x_2^2\rvert))^2 - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$
12	Trid 6	$f(x) = \sum_{i=1}^{D}(x_i - 1)^2 - \sum_{i=1}^{D} x_ix_i - 1$
13	Trid 10	$f(x) = \sum_{i=1}^{D}(x_i - 1)^2 - \sum_{i=1}^{D} x_ix_i - 1$
14	Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + (\frac{1}{2}\sum_{i=1}^{n} ix_i)^2 + (\frac{1}{2}\sum_{i=1}^{n} ix_i)^4$
15	Powell sum	$f(x) = \sum_{i=1}^{D}\lvert x_i\rvert^{i+1}$
16	Quartic	$f(x) = \sum_{i=1}^{n} ix_i^4 + random(0,1)$
17	Schwefel 2.20	$f(x) = -\sum_{i=1}^{n}\lvert x_i\rvert$
18	Schwefel 2.21	$f(x) = max\lvert x_i\rvert$
19	Schwefel 2.22	$f(x) = \sum_{i=1}^{n}\lvert x_i\rvert + \Pi\lvert x_i\rvert$
20	Schwefel 2.23	$f(x) = \sum_{i=1}^{n} {x_i}^{10}$
21	sphere	$f(x) = \sum_{i=1}^{D} {x_i}_i^2$
22	step	$f(x) = \sum_{i=1}^{D}\lvert x_i\rvert$
23	Step No.2	$f(x) = \sum_{i=1}^{D}(x_i + 0.5)^2$
24	Stepint	$f(x) = 25 + \sum_{i=1}^{D}\lvert x_i\rvert$
25	Sumsquares	$f(x) = \sum_{i=1}^{D} ix_i^2$
26	Ackley	$f(x) = -20e^{-0.02\sqrt{D^{-1}\sum_{i=1}^{D} x_1^2}} - e^{D^{-1}\sum_{i=1}^{D}\cos(2\pi x_i)} + 20 + e$
27	Bird	$f(x) = \sin(x_1)\, e^{(1-\cos(x_2))^2} + \cos(x_2)\, e^{(1-\sin(x_1))^2} + (x_1 - x_2)$
28	Bohachevsky No.2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\,.\,0.4\cos(4\pi x_1) + 0.3$
29	Bohachevsky No.3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_1) + 0.3$
30	Camel Six-Hump	$f(x) = 2x_1^2 + 1.05x_2^4 - \frac{x_1^6}{6} + x_1x_2 + x_2^2$
31	Cross in	$f(x) = -0.0001[\sin(\mathrm{x}_1)\sin(x_2)\ e^{\lvert 100 - [x_1^2 + x_1^2]/\pi\rvert} + 1]^{0.1}$
32	Drop Wave	$f(x) = -\frac{1+\cos(12\sqrt{x_1^2 + x_2^2})}{0.5\,(x_1^2 + x_2^2) + 2}$
33	Egg Holder	$f(x) = \sum_{i=1}^{m-1} -(x_{i+1} + 47)sin\sqrt{\left\lvert x_{i+1} + \frac{x_i}{2} + 47\right\rvert} - x_i sin\sqrt{\lvert x_{i+1} + 47\rvert}$
34	Griewank	$f(x) = \sum_{i=1}^{n} x_i^2/4000 - \Pi\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
35	GoldStein-Price	$f(x) = [1 + (x_1 + x_{2+1})^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [+(2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2$
36	Hartman 3	$f(x) = -\sum_{i=1}^{4} c_i\, exp\left[-\sum_{j=I}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right]$

Table 3.2 The Equations for each benchmark test functions

f_n	Test	Objective Function
37	Hartman 6	$f(x) = -\sum_{i=1}^{4} c_i\, exp\left[-\sum_{j=I}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right]$
38	Holder-Table	$f(x) = -\left\|\cos(x_1)\cos(x_2)e^{\|1-(x_1+x_2)^{0.5}/\pi\|}\right\|$
39	Keane	$f(x) = \frac{sin^2(x_1-x_2)sin^2(x_1-x_2)}{\sqrt{x_1^2+x_2^2}}$
40	Levy No.13	$f(x) = sin^2(3\pi x_1) + (x_1-1)^2[1+\sin^2(3\pi x_2)] + (x_2-1)^2\ [1+ \sin^2(3\pi x_2)]$
41	Penalized	$f(x)\ \frac{\pi}{n} \times \{ 10\ sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})] + (y_n-1)^2\} + \sum_{i=1}^{n} u(x_i,a,k,m)$ $whee$ $y_i 1 + \frac{1}{4}(x_i+1)$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & if\ x_i > a \\ 0 & if -a \leq x_i \leq a \\ k(-x_i-a)^m & if\ x_i < -a \end{cases}$
42	Penalized N.2	$f(x) = 0.1 \times \{ sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i-1)^2[1+10sin^2(3\pi x_{i+1})] + (x_n-1)^2[1+sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i,a,k,m)$ $where$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m & if\ x_i > a \\ 0 & if -a \leq x_i \leq a \\ k(-x_i-a)^m & if\ x_i < -a \end{cases}$
43	Perm	$f(x) = \sum_{i=1}^{d}(\sum_{j=1}^{d}(j^i+\beta)((\frac{x_j}{j})^i - 1))^2$
44	Powersum	$f(x) = \sum_{i=1}^{d}[(\sum_{j=1}^{d} x_j^i) - b_i]^2$
45	Price 1	$f(x) = (\|x_1\| - 5)^2 + (\|x_2\| - 5)^2$
46	Price 2	$f(x) = 1 + sin^2 x_1 + sin^2 x_2 - 0.1e^{x_1^2 - x_2^2}$
47	Price 3	$f(x) = 100(x_2 - x_1^2)^2 + 6[6.4(x_2 - 0.5)^2 - x_1 - 0.6]^2$
48	Price 4	$f(x) = (2x_1^3\ x_2 - x_2^3)^2 + (6x_1 - x_2^2 + x_2)^2$
49	Shubert	$f(x) = \prod_{i=1}^{n}\sum_{j=1}^{5}\cos((j+1)\,x_i + j)$
50	Shubert No.3	$f(x) = \sum_{i=1}^{D}\sum_{j=1}^{5} jsin\ ((j+1)\,x_i + j)$
51	Shubert No.4	$f(x) = \sum_{i=1}^{D}\sum_{j=1}^{5} jcos\ ((j+1)\,x_i + j)$
52	Alpine	$f(x) = \sum_{i=1}^{D}\|x_i \sin(x_i) + 0.1x_i\|$
53	Bohachevsky	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(2\pi x1) - 0.4\cos(4\pi x2)$
54	Booth	$f(x) = (x_2 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
55	Branin	$f(x) = (x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$
56	Egg Crate	$f(x) = x_1^2 + 2x_2^2 + 25(sin^2(x_1) + sin^2(x_2))$
57	Michalewicz2	$f(x) = -\sum_{i=1}^{2}\sin(x_i)\, sin^{2m}(\frac{ix_i^2}{\pi})$
58	Michalewicz5	$f(x) = -\sum_{i=1}^{5}\sin(x_i)\, sin^{2m}(\frac{ix_i^2}{\pi})$
59	Michalewicz10	$f(x) = -\sum_{i=1}^{10}\sin(x_i)\, sin^{2m}(\frac{ix_i^2}{\pi})$
60	Rastrigin	$f(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10\cos(2\pi x_i)]$

3.4 Experimental Settings

The performance of the algorithm was evaluated by carrying out 2 sets of experiments. The first experimental set looked into the overall performance of the algorithms over a fixed number of iterations. Upon completing certain number of iterations, the performance of the algorithms was evaluated based on the mean and the best fitness values found for each benchmarked function. The number of iterations employed in this study had been fixed at 1000. Next, the second experimental set investigated the convergence behaviour of the algorithms. In this case, the algorithm was run on various numbers of iterations to evaluate the mean fitness value established for each case. Hence, the convergence behaviour of the algorithms based on the number of iterations was obtained. The proposed NPO algorithm in this study was applied on new combination of benchmarked functions, while its performance was compared to that of five well-known algorithms (PSO2011, ABC, GWO, FPA, and FFA) (Clerc, 2011; Karaboga and Basturk, 2007; Mirjalili et al., 2014; Yang, 2008, 2012).

3.5 Summary

In summary, this chapter presented the research methodology of the thesis. The methodology of this research consists of three main stages, Literature review stage, Methodology stage, and results and discussion stage. In first stage, the gap and issues have been analyzed and discussed, while in the second stage, the proposed algorithm (i.e., NPO) is designed and proposed. Finally, the third stage presents the experimental settings and the benchmark test function used in this study for evaluating the proposed algorithm in terms of finding the best solutions, the scalability, and the exploration and exploitation rates. Next chapter, the source of inspiration with the mathematical model of NPO is explained in details.

CHAPTER 4

NOMADIC PEOPLE OPTIMIZER

4.1 Introduction

In the previous chapter, the theoretical background and mathematical notation have been presented. The main concepts on optimization problems have been elaborated. The chapter concluded that there is always a need to develop a new metaheuristic, due to the issues faced by the current state of art metaheuristics. In addition, No Free Lunch theorem logically proved that no one can propose an algorithm for solving all optimization problems. This means that the success of an algorithm in solving a specific set of problems does not guarantee solving all optimization problems with different type and nature. The NFL theorem allows researchers to propose new optimization algorithms or improve/modify the current ones for solving subsets of problems in different fields.

In this chapter, novel metaheuristic is proposed and discussed. The proposed algorithm is called “Nomadic People Optimizer (NPO)”. The proposed metaheuristic contains a novel multi-swarm cooperative approach, called “Meeting Room Approach (MRA)”. MRA simulates the human interaction when finding solutions to life problems. The chapter is divided into two main parts, first part presents the source of inspiration. While the second part presents the mathematical model and the main steps of NPO. The last step is called “Periodical Meeting” represents the MRA.

4.2 Source of Inspiration

4.2.1 Nomadic People

Nomads refer to those who live their entire life traveling from one place to another with their herds of camels, cattle, and sheep in search of natural sources of water and food. These herds graze on pastures close to water sources and provide their owners food, as well as other major necessities, such as skin and wool for clothing and tent-making. The milk from the herd serves as a source of calcium and protein for the nomads. It is well-known that nomads do not familiarize themselves with an environment or cultivate the lands within their settlement as they do not settle in one place for a long time.

Early humans were also nomadic as they lived in places with available trees or animals for hunting. Their survival in these territories depends on food sources. Upon depletion of water source, they immediately travel to another place to settle for another period. Therefore, they live a life of constant traveling and do not have permanent settlements. Nomads are still found in several regions across the globe, mostly in Africa, Asia, South America, Mexico, and the Middle East. Their lifestyle, in general, still remains a life of traveling from place-to-place in search of food and water, but over time, they have eventually become more civilized, when compared to the early nomads. At present times, they travel in vehicles to ease movement of herds and transportation of luggage. In fact, the nomads can be categorized into several clans, such as the Berbers, the Gypsy, and the Bedouins. The suggested algorithm in this thesis mainly resembles the lifestyle of the Bedouins.

4.2.2 The Bedouins

The Bedouins are the most popular type of nomads whose lives depend completely on the principle of desertion. They only travel in the desert in search of water and pasture. The Bedouin clans are found in the deserts of the Arabian Peninsula, Western Iraq, parts of Syria, Jordan, Palestine, Egypt and North Africa. The word “Bedouin” came from the Arabic word “Badawi” (for the singular) and “Badu” (for the plural). The Bedouins, in general, are Arabic clans who live in the deserts of the Arabic countries and North Africa. Figure 4.1 shows the distribution map of the Bedouins.

The Bedouins mainly rely on their grazing animals for their lives. The most prominent of these animals are the camels because they use them to transport their tents when moving from place to place. They also use the skin of the camels to make their tents. Other animals such as sheep and goats serve as a daily source of meat and milk for the nomads and their families. These animals are constantly provided with steady sources of food and water to ensure a continuous supply of food and other daily needs for the nomad. This is the reason for the consistent exploratory life of the nomads and their animals. They also trade their livestock to earn some money for their daily up-keep.

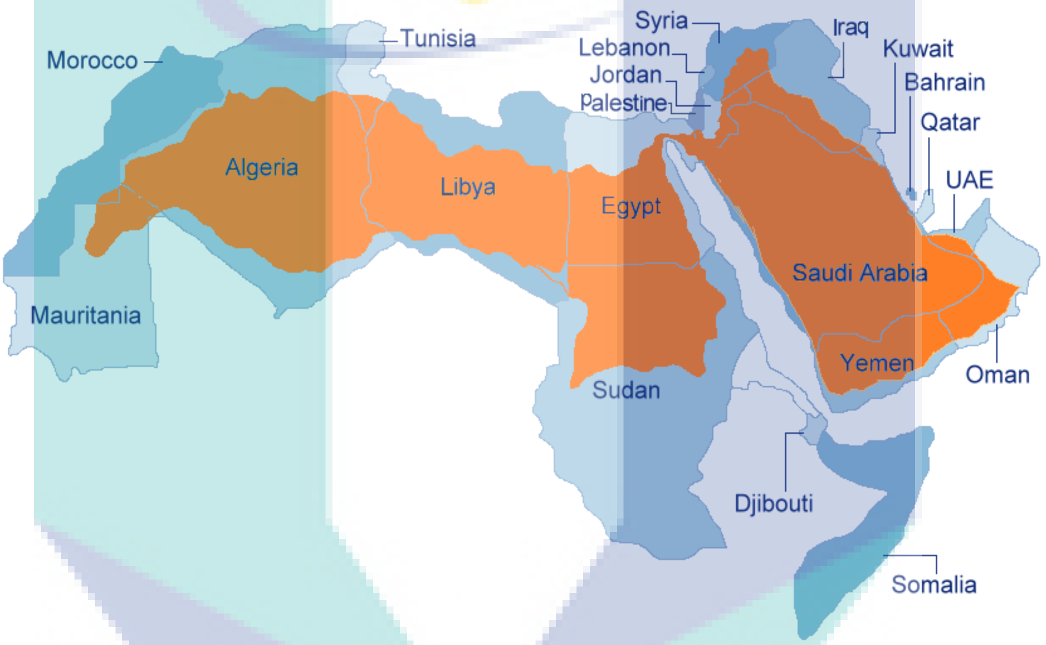


Figure 4.1 The distribution of the Bedouins over Arabic countries

The Bedouin nomads live in clans and each clan has a leader called a Sheikh in the Arabic language. The Sheikh controls the activities of the members of his clan; he decides when they should migrate to another place, and responsible for solving both internal and external problems within his clan. The position of a Sheikh is usually hereditary but never through an election (i.e. sometimes revolves within the clan). However, the Sheikh must command the strongest influence within the clan in terms of affluence (may be the number of livestock or the wealth acquired from trading livestock). Thus, the Bedouin community is a tribal society subject to the tribal laws established by their ancestral communities. They are not subject to the laws of the state or the country they reside. Generally, they are well separated from the happenings within the country as they are more interested in their own internal affairs than the affairs of their surroundings.

The Bedouins, as earlier explained, live in a tribal society. They have age-long customs, traditions, and rules that govern them and any violation of these rules and traditions attracts severe punishments or even expulsion from the clan. Their adherence to their customs and traditions has enabled their survival despite the difficult nature of their lives. These laws and customs are enacted by the leaders of the clans who usually meet to discuss external problems, such as when the grazing season should begin. They also meet to help each other in solving certain problems or even to consolidate relationships among themselves (through marriages). Such meetings usually take place in the tent of one of the leaders or sometimes in the tent of the most influential or oldest Sheikh.

On the other hand, there is a high chance of wars between the clans around the crops areas and water sources despite the peaceful meetings due to the harsh nature of living in the desert. Each clan is struggling to survive in the desert, and thus, can enter wars with other clans to preserve the sources of life. The members of each clan are always prepared for war whenever it ensues due to the earlier stated reasons. These wars and the consistent life of traveling from place to place have made them physically well-built. However, the clans can settle their scores either by peaceful meetings or engage in wars and this is an important point in the NPO algorithm which will be clarified later.

Regarding their way of living, the Bedouin have no fixed place to live. They are totally dependent on the tents (also called a house of fur) woven from the fur of black goats and in some of its sections, from a mixture of furs from camels and goats for decoration. Wood or shrubs are used to lift the tents from the ground. The tents offer many important features for a nomadic life in general and the Bedouins in particular. For instance, they can be easily folded when moving from place to place, and are suitable for a comfortable living during the climate changes between summer and winter or between night and day. On the other hand, the cost of tent maintenance is much less than the cost of maintaining regular houses built with solid materials. The women among the Bedouins are mainly responsible for the cutting of wools and furs, as well as spinning and knitting the tents and carpets. These activities are done inside the existing tents when establishing new ones.

As earlier stated, the leader of each clan (Sheikh) fully controls the members of his clan; he decides where each family should to establish their tent. Generally, these families are meant to establish their tents in a semi-circular manner around the tent of the Sheikh. Figure 4.2 illustrates the distribution of family tents by the Sheikh of a clan.

4.2.3 Observations and Rules

Based on the nature and the lifestyle of the Bedouins, two primary observations can be made, which happen to inspire the new algorithm; NPO. The first observation refers to the classification of their families. There are two classes of their families, in which the first class is the Sheikh family. The Sheikh family is responsible for several fateful decisions, such as the time to move from a present place to another. They also determine the location of the new place to move to, as well as the pattern of distributing the families around the Sheikh's tent in the clan. Meanwhile, the second class is made up of the families of the rest of the clan (i.e. normal families). These families possess no power over the decisions made by the Sheikh family, but obey orders from the Sheikh. The second observation is regarding their lifestyle. In general, several rules have been outlined, as follows:

i) The Bedouins live a life of continuous traveling in search for suitable places to live in, depending on the availability of sources of water, plantation, and food for them, as well as for their livestock.

ii) The clans either peacefully meet each order for deliberations or engage in bloody fights to overcome their variances. A peaceful meeting can be summoned to encourage marriages between two clans, as well as to reduce risk of conflicts between them. On the other hand, war may spark between clans as a Sheikh attempts to take over a region with better sources of water and grazing field.

iii) The Sheikh is the supreme authority who decides the fate of the families and their members; he decides either to fight a war with another clan or to join in peaceful meetings. He also decides when and where the clan should move.

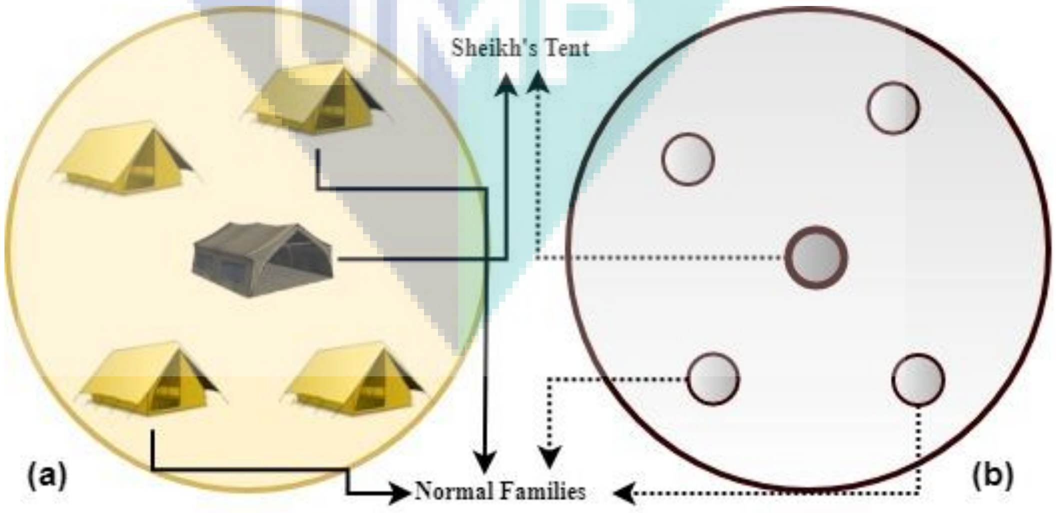
iv) The Bedouins use their tents as their main housing due to several advantages. A tent is provided for each family, and these tents are distributed in a semi-circular pattern, with the tent of the Sheikh at the center. Figure 4.2 portrays the distribution of tents around a Sheikh's tent.

v) The authority of the Sheikh over the rest of the clan is not conferred via democratic process, but elections are held in rare cases. Commonly, the transition of leadership is either hereditary (from father to son) or in situations of conflict where a normal family may become more influential than the Sheikh's, thus resulting in taking over power from the Sheikh's family.

vi) The Sheikh sends the families to scout for good places for living. These families move randomly in different directions and distances. When a family finds a better place than the present one, the Sheikh moves towards the new position and re-establishes the clan (i.e. the normal families) in a semi-circular shape around his tent.

vii) The annual migration of the Bedouins usually occurs during the summer and winter seasons, exploiting the slight climatic and territorial changes by using their seasonal or periodic movements between the summer and winter pasture areas (SPA and WPA). Typically, the location of the SPA is usually determined by the availability of water and pasture sources, as well as suitable climatic conditions. The WPA, on the other hand, aims for places (depressions surrounded by sand dunes) with wells and dams, which could be either small or closed areas. They occupy the SPA between the period of May and October (late harvest period) and move to the WPA for the rest of the year.

The Bedouins live an exciting and inspirational lifestyle, thus the NPO algorithm is inspired by their way of life. The mathematical model of the NPO is depicted in the following subsection.



(a) Families in the clan **(b)** Solutions in the search space
Figure 4.2 Semi-circular distribution of the families

4.3 Mathematical Model of NPO

The unique lifestyle of the Nomads, in general, and Bedouins, in particular, has enabled their settlement in the desert for centuries. This unique lifestyle has inspired the development of a new metaheuristic that may lead to viable solutions for optimization problems. Table 4.1 presents the list of variables employed to develop the mathematical model of the NPO algorithm. The NPO algorithm is comprised of five main steps, which are 1) initial meeting, 2) semi-circular distribution, 3) families searching, 4) leadership transition, and lastly, 5) periodical meeting.

Table 4.1 List of variables used in NPO

Symbol	Meaning	Range
σ	Sheikh of the Clan	Integer
$\overrightarrow{\sigma_c}$	The position of the Sheikh	Integer
UB	Upper Bound	Based on the Problem
LB	Lower Bound	Based on the Problem
$Rand$	Random Number between 0 and 1	[0,1]
Rd	Reduce of the circle	UB / 2
θ	The value of the angle	$0 – 2\pi$
Ψ	The direction variable	1 , -1

The terminologies used to describe NPO are discussed below:

1. Sheikh (σ): Leader, an individual represents the current local best solution in the swarm.
2. Best Leader (σ_B): An individual represents the global best solution in all swarms, which is used in the meeting room approach.
3. Normal Leader (σ_N): An individual represents the other leaders except the Best Leader (σ_B).
4. Family (x): An individual represents a member in the swarm or clan which has a lower fitness value than the leader.
5. Clan (c): a group of families (x), including the Leader (σ), which represents an individual swarm. NPO consists of several clans, each clan consists of several families and single Leader.
6. Fitness: a term refers to the function or method to evaluate the goodness of a position in the search space. It takes the coordinates in the solution space and returns a numerical value (goodness). The fitness function provides an interface between the physical problem and the optimization algorithm.
7. Direction (Ψ): It is a variable used for guiding the Normal Leaders towards the Best Leaders.

Step 1: Initialize all leaders in the algorithm. They meet only to distribute the families of their clans in the desert. The location of each Sheikh is randomly determined using following equation:

$$\overrightarrow{\sigma_c} = (UB - LB) \times Rand + LB \qquad 4.1$$

where UB, LB represent the upper bound and lower bound respectively, while Rand denotes a random value between 0 and 1, and $\overrightarrow{\sigma_c}$ represents the position of the Sheikh of the clan c

Step 2: Create an individual clan (a set of families) for each leader, by distributing the families around their leader in a semi-circular shape.

After a Sheikh has moved to his position in the desert (the search space), he decides based on his authority the location of each family around his own tent in a semi-circular shape, as well as the position of his own tent at the centre of the circle. Figure 3 illustrated the semi-circular distribution of the families around the Sheikh's tent.

Mathematically, it is possible to distribute points randomly within a given circle with a known radius using the equation of the 2D circle. These points are circled around the origin (centre of the circle) by the value of the angle, as given in Equations

$$X = \left(Rd \times \sqrt{R_1}\right) \times \cos(\theta) + X_0 \qquad 4.2$$

$$Y = \left(Rd \times \sqrt{R_2}\right) \times \sin(\theta) + Y_0 \qquad 4.3$$

where X_0 and Y_0 represent the coordinates of the origin point (centre of the circle), while R_1 and R_2 denote the random coordinates of a point within the perimeter of that circle. Meanwhile, θ refers to the angle value of that point, which is a random value lies between $[0, 2\pi]$.

Equations (4.2) and (4.3) are used when the generated points are within a circle in 2D shape, (i.e., X, Y, and θ). Nevertheless, if the solutions are represented within the search space, the problems do not require any X and Y coordinates. Hence, the representation of the solutions is unary (i.e. single dimension), instead of 2D. As such, the distribution of tents randomly around the Sheikh's tent requires an X coordinate, while excluding the non-required Y coordinate. With that, the equation was developed to fit this scenario, as given in Equation (3.4) in the following:

$$\overrightarrow{x_i^c} = \overrightarrow{\sigma_c} \times \sqrt{Rand} \times \cos(\theta) \qquad 4.4$$

where $\overrightarrow{x_i^c}$ represents the position of a family, $\overrightarrow{\sigma_c}$ represents the position of the Sheikhs' tent. It can be noted from equation (4.4) that the value of the family location is based entirely on the value of the location of the Sheikh, and this is within the powers of the Sheikh since he is in charge of distributing the families around his tent. The families and the clans were represented as vectors because they are not a single value but represent the values of the solutions which are not less than one value and no more than the number of dimensions of the problem itself.

While this step can be used to initiate the normal families around the Sheikh, it also can be used as a local search process for it depends on two random values ($Rand$ and θ). Therefore, this step can be called at any time, mainly because it has an effect on the positions of the normal families, apart from enhancing the convergence of the algorithm. Figure 4.2 shows the random distribution of normal families in the desert and in the search space as well.

Step 3: For each family in each clan, explore the search space and find better positions. This step is used when the source of food is not enough. In such situations, the Sheikh deploys each family to find a better source of food, any family that finds a better source of food will assume authority over that location and becomes the new Sheikh. In our algorithm, the families can move in different directions in the search space based on random steps and directions generated by the Levy Flight formula as follows:

$$\overrightarrow{x_{new}^c} = \overrightarrow{x_{old}^c} + (a_c * \overrightarrow{(\sigma_c - x_{old}^c)} \oplus Levy) \qquad 4.5$$

where $\overrightarrow{X_f^{new}} and \overrightarrow{X_f^{old}}$ represent the new and old positions of the current family respectively, a_c represent the area of the clan which is the average distance between all the normal families and the Sheikh's tent. a_c can be calculated using the following equation: -

$$a_c = \frac{\sum_{i=1}^{\#noF} \sqrt{(\sigma_c - x_i^c)^2}}{\#noF} \qquad 4.6$$

The distance between the normal families and the Sheikh's family gets closer for good, thus, the value of a_c enhances the search process when its value is decreased. The families move in different directions, and in random step sizes; the step sizes are generated by the Levy flight (λ_c) equation as follows:

$$Levy \sim u = t^{-\lambda} \quad (1 < \lambda \leq 3) \qquad 4.7$$

The Levy flight equation is usually used to generate a random walk while drawing the random step length from a Levy distribution with an infinite mean and variance(Yang and Deb, 2009a). The stochastic equation for random walk is typically represented in Equation (4.5). A random walk is generally a Markov chain that depends on its current location (the first term in the above equation) and the transition probability (the second term) to determine its next status/location. The product $\oplus$ means entry-wise multiplications. Here, the random walk via Levy flight is more efficient in exploring the search space as its step length is much longer in the long run.

Step 4: For each clan, swap the new current best solutions with the current leader. As explained earlier, the leadership amidst the Bedouins is either hereditary (from father to son) or through revolutions, which never involved the democratic process. A family may have control over a water source and thus, expand its influence and control over the rest of the families. Therefore, the power of the former Sheikh's family becomes weaken, hence allowing the new family to take over the leadership of the clan. This feature is a crucial factor in the NPO algorithm as it enables the state of solutions to be enhanced from the worst to better. In precise, it represents the Local Best Solution, which in turn, reflects the Exploitation side of the algorithm. When a family discovers a better solution value, it becomes the new Sheikh of that generation.

Step 5: Determine the best leader ever over all leaders, then, move the other normal leaders towards the best leader ever. It is a periodical meeting between the best solutions which is performed within each iteration.

The periodical meetings are dissimilar to the initial meeting (i.e. step 1), except for the redistribution of Sheikhs in the desert. During these periodical meetings, the

Sheikhs strive to resolve any external problem and discuss the best locations for relocation purpose. The reason for this meeting is to enable each Sheikh to have control over his place, but without arousing the ambitions of others, instead, bringing them closer to himself.

The periodic meetings occur in two stages and they involve only the Sheikhs. The families are disallowed from interfering, except those in power. The first phase of the meeting is to determine the most powerful Sheikh, or in precise, the Sheikh of the best location who will propose solutions to other Sheikhs for them to update their locations. This update is performed by adding the variance between the position of the strongest Sheikh and that of the normal Sheikh as depicted in the following equation:

$$\Delta Pos \ = \Psi \left(\frac{\sqrt{\sum_i^D (\sigma^E - \ \sigma_c^N)^2}}{\#D} \right) \qquad 4.8$$

where σ^E represents the position of the best Sheikh, and σ_{Ni} denotes the position of the normal Sheikhs. Meanwhile, $\#D$ is the number of dimensions of the problem, Ψ refers to the direction, and ΔPos represents the normalized distance between the best sheikh and the normal sheikh. The direction variable Ψ guides the normal sheikhs to better positions depending on the fitness value of the best sheikh, as follows:

$$\Psi = \begin{cases} \ \ 1 & if\ \mathrm{f}(\sigma^{E}) \geq 0 \\ -1 & otherwise \end{cases} \qquad 4.9$$

The normal sheikhs update their positions via equation (4.10). This equation represents the exploration stage in NPO.

$$\overrightarrow{\sigma_c^{new}} = \ \overrightarrow{\sigma_c^N} + \ \Delta Pos\ (\sigma^E - \ \sigma_c^N) * \frac{IT}{\#T} \qquad 4.10$$

where $\sigma_c^{new} and\ \sigma_c^N$ represent the new and old position of the normal sheikh respectively, while IT and $\#T$ represent the current iteration and the total number of iterations respectively.

During the periodical meeting, the positions of all normal sheikhs are updated. The sheikh stays at the new position if it is better than before, apart from establishing his new clan based on the second step (semi-circular distribution), otherwise he returns to the old position. It is important to mention that the periodical meeting is a unique method of sharing information between swarms, for it reflects a cooperative scheme for multi-swarms. As mentioned before, each clan represents an individual swarm, while the periodical meeting facilitates communication between them.

This cooperative multi-swarm scheme is called Meeting Room Approach (MRA). Figure 4.3 illustrates the structure and the block diagram of MRA. MRA can be applied with other metaheuristics, such as PSO, FA, and BA. It helps the algorithms to balance between exploration and exploitation, which promotes faster convergence, in comparison to other standard versions of the algorithms.

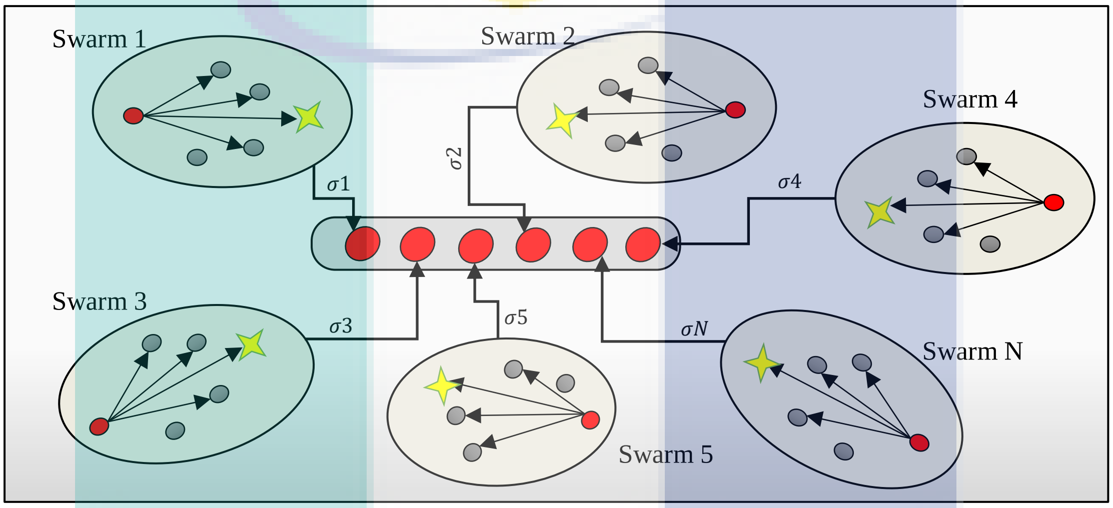


Figure 4.3 Meeting Room Approach

The pseudocode of MRA is given in Figure 4.4, while the pseudocode and flowchart of NPO are given in Figure 4.5 and Figure 4.6 respectively.

Algorithm : Meeting Room Approach

```
1.   Input: All Sheikhs – or Leaders – in the swarms
2.   Output: Best Sheikh Ever, Updated Positions for all Normal Sheikhs
3.   Procedure:
4.   $\sigma^B$ = Determine the best Sheikh
5.   For $i = 1$ To $\#Sheikhs - 1$
6.       Calculate $\Delta Pos$ between $\sigma_i^n$ and $\sigma^B$ via $eq. 4.8$
7.       Update the position of $\sigma_i^n$ via $eq. 4.10$
8.       Re-initialize the $Clan_i$ – or $swarm_i$ – based on the new position of $\sigma_i^n$
9.   End For
10.  Return the best Sheikh ever
```

Figure 4.4 Pseudocode of MRA

Algorithm : Nomadic People Optimizer (NPO)

```
1.  Input: No. of Clans (#Clans), No. of Families (#F), No. of Iterations (#$T$)
2.  Output: The Best Sheikh
3.  Procedure:
4.  Define the objective function $f(x)$,
5.  Initialize the Leaders $\sigma_c^O$ $, c = \{1, 2, 3, \ldots\ , No.of\ Clans\}$ as follows:
            $\sigma_c^O = (UB - LB) \times Rand + LB$
6.  Calculate the fitness value for each leader using the objective function
7.  Repeat $(Itr)$:
8.    For $c$ = 1 to $\#Clan$
9.      Distribute the solutions around the leader in a semi-circular shaper as follows:
            $\overrightarrow{x_i^c} = \overrightarrow{\sigma_c} \times \sqrt{Rand} \times \cos(\theta)$
10.     Calculate the fitness value for each solution $x_i^c$ using the objective function
11.     Set the best $x_i^c$ in the clan $c$ as $\sigma_c^B$
12.       If $\sigma_c^B$ is better than the original $\sigma_c^O$ Then, swap them $\sigma_c^O = \sigma_c^B$
13.       Else: Explore the search space using the following steps:
14.         Calculate the average distance between all families as follows:
            $a_c = \frac{\sum_{i=1}^{\#noF} \sqrt{(\sigma_c - x_i^c)^2}}{\#noF}$
15.         Move the family towards the new position:
            $\overrightarrow{x_{new}^c} = \overrightarrow{x_{old}^c} + (a_c * \overrightarrow{(\sigma_c - x_{old}^c)} \oplus Levy)$
16.         Calculate the fitness value for each solution $x_i^c$ using the objective function
17.         Set the best $x_i^c$ in the clan $c$ as $\sigma_c^B$
18.       End if
19.   End For
20.   Determine the best leader ever as $\sigma^E$
21.   Determine the value of the direction variable $\Psi$ as follows:
            $\Psi = \begin{cases} 1 & if\ (\sigma^E) \geq 0 \\ -1 & otherwise \end{cases}$
22.   Calculate $\Delta Pos$ via the following function
            $\Delta Pos = \Psi \left( \frac{\sqrt{\sum_i^D (\sigma^E - \sigma_c^N)^2}}{\#D} \right)$
23.   For each normal leader $\sigma_c^N$, move towards the best leader ever $\sigma^E$, as follows:
            $\overrightarrow{\sigma_c^{new}} = \overrightarrow{\sigma_c^N} + \Delta Pos\ (\sigma^E - \sigma_c^N) * \frac{IT}{\#T}$
24.         Calculate the fitness value for each $\sigma_c^{new}$ using the objective function
25.         If: the $\sigma_c^{new}$ is better than the $\sigma_c^N$, Then keep it
26.         Else: keep the $\sigma_c^N$
27.   End For
28. Loop Until $(Itr > \#T)$
29. Return $\sigma^E$
```

Figure 4.5 Pseudocode of NPO

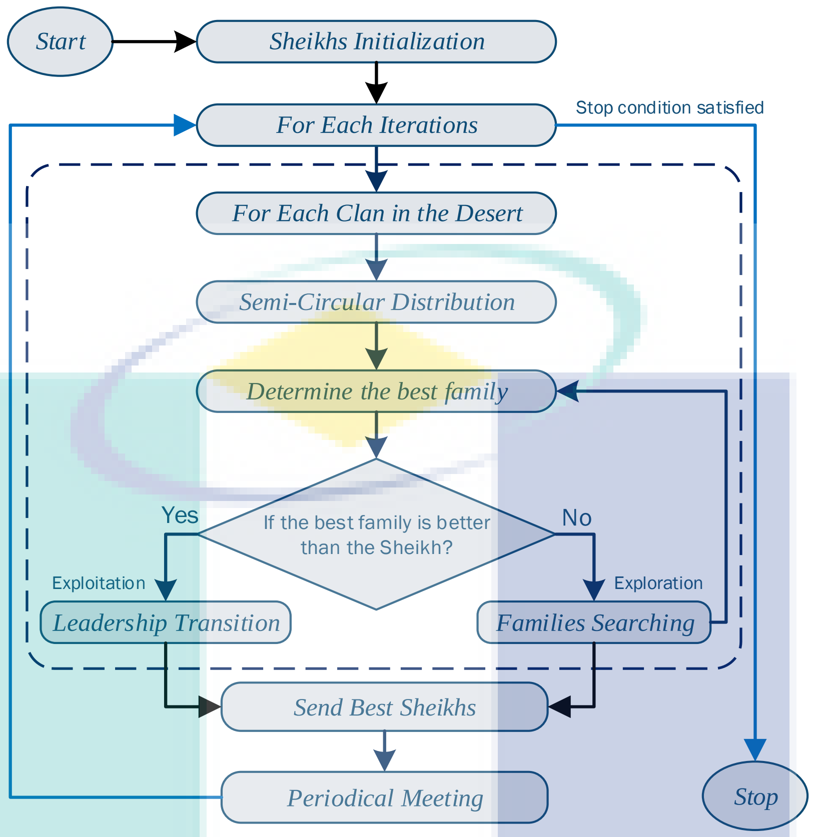


Figure 4.6 Flowchart of NPO

The time complexity of NPO has been calculated, it is equal to:

$$O\big(TC(F-1)\big) \tag{4.11}$$

where $\#T$ represents the total number of iterations, while C represents the number of clans, F represents the number of Families. The first part of the equation above contains three main loops, the main loop (T), the inner loop (C), and the loop inside the familis searching step (F). This step needs F-1 loops for execution, because the Sheikh of that clan does not move or search for a new positon. The second part contains the time of MRA, which represents the time complexity of the meeting room approach procedure, $\#\sigma$ represents the number of Sheikhs in the meeting room, which is equal to C in the previous equation.

4.4 Graphical Illustration of NPO

As earlier stated in the previous section, the proposed NPO has two exploitation steps and two exploration steps. The final step is used to achieve a balance between exploitation and exploration. In this section, both exploration and exploitation are simulated in figures for a better understanding. The last section provides an analysis of the balancing mechanism.

4.4.1 Exploitation

The semi-circular distribution step helps the Sheikhs to establish their clans (i.e. normal families) based on equation (4.4). This step represents a local search as well. The leadership transition step changes or swaps Sheikhs with the best family in the clan. Both represent the local search algorithm or exploitation. It is worth mentioning that NPO does not implement a traditional local search algorithm, in other words, the semi-circular distribution employs a re-initialization method for the normal families based on the position of the Sheikh's tent. Figure 4.7A and Figure 4.7B illustrate the effect of these two steps on the search process.

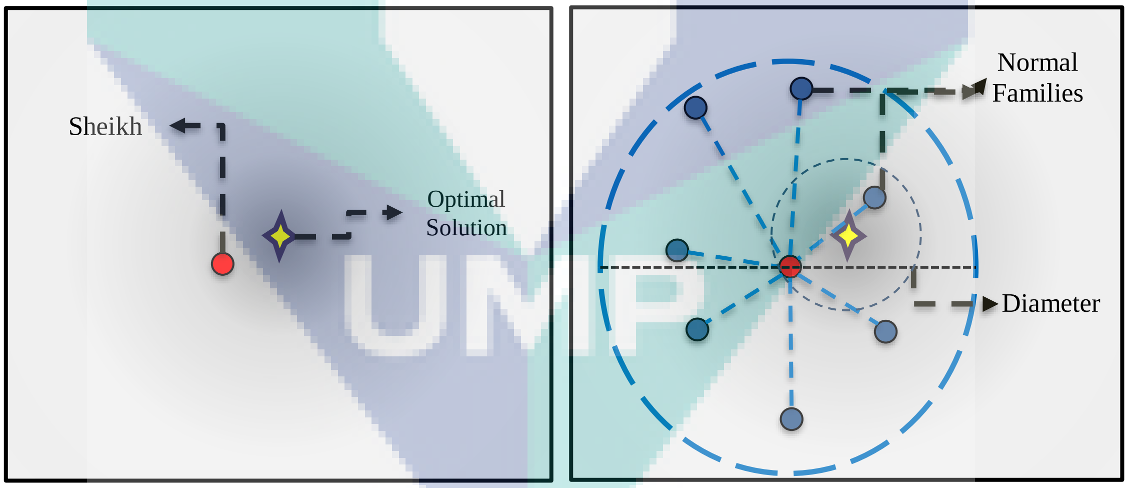


Figure 4-7A Initialization (One Sheikh) Figure 4-7B Semi-Circular Distribution

Figure 4.7 Illustration of the first two steps of NPO

Figure 4.7A illustrates the first step of the NPO algorithm. The leader is initialized in the search space by using the first step which is represented by the red circle. In Figure 4.7B, all the families are initialized around the leader, in other words, the clan is established by using the second step.

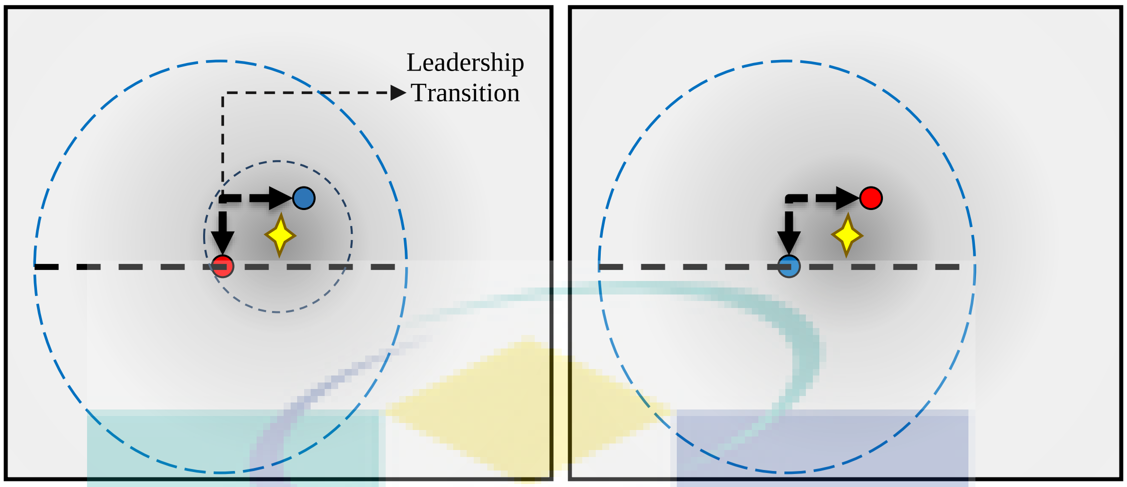


Figure 4.8A Before Leadership Transition Figure4.8B After Leadership Transition
Figure 4.8 Illustration of the exploitation step of NPO

In Figure 4.8A, a family inside the clan has a better place – or position – than the leader, for this reason, this family got the leadership from the leader and became the new leader of that clan. Figure 4.8B illustrates that the previous leader became a normal family (blue circle) while the family became the new leader (red circle).

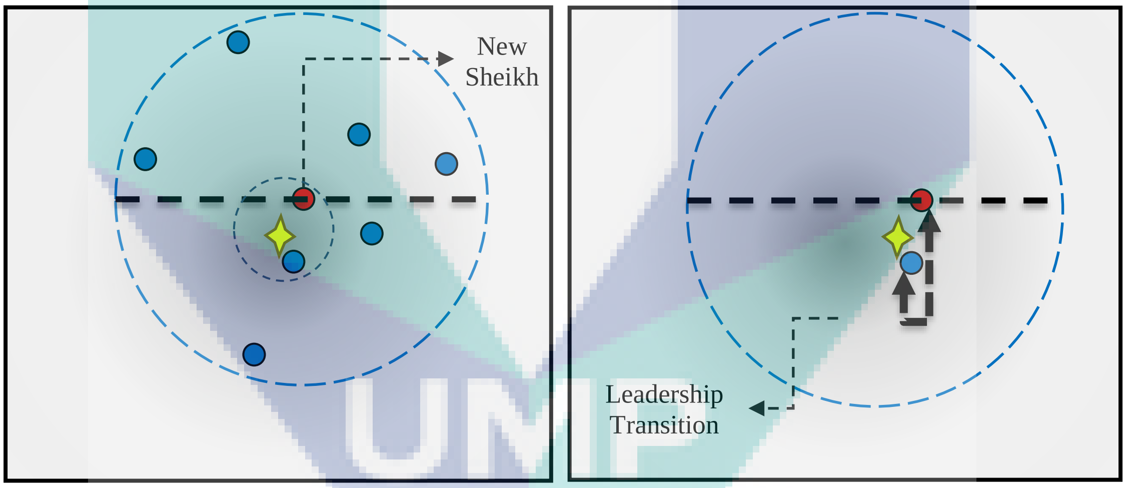
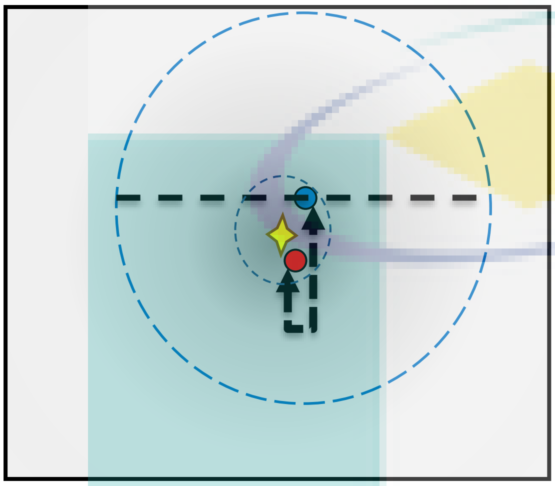


Figure 4.9A Semi-Circular Distribution Figure4.9B Leadership Transition
Figure 4.9 Illustration of the exploitation steps of NPO

The new leader established the new clan around him using the second step in Figure 4.9A. Then, one of the new families has a better position – or near to the optimal solution – which meaning that there is a new leader to the clan. This process represents the local search part of NPO, by searching around the current local best solution. The rest of figures in this section represent the local search of NPO until the stop condition is satisfied.

It is worth to mention that this simulation illustrates the searching process of only single clan – or swarm – while in the NPO, there are C clans are searching at the same time. Meaning that, the local search is performed multiple times in different places in the search space, and there are several local best solution for each clan.



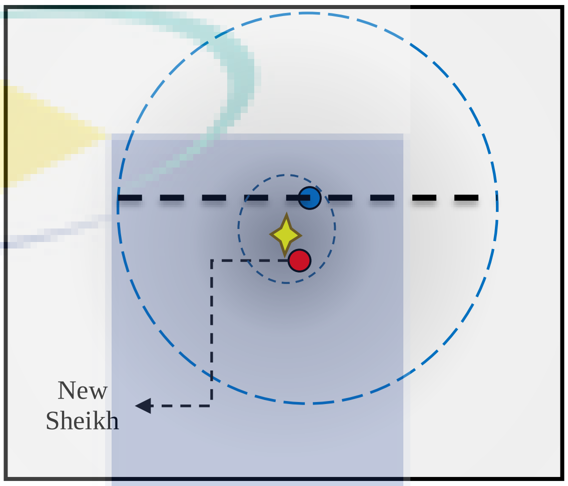
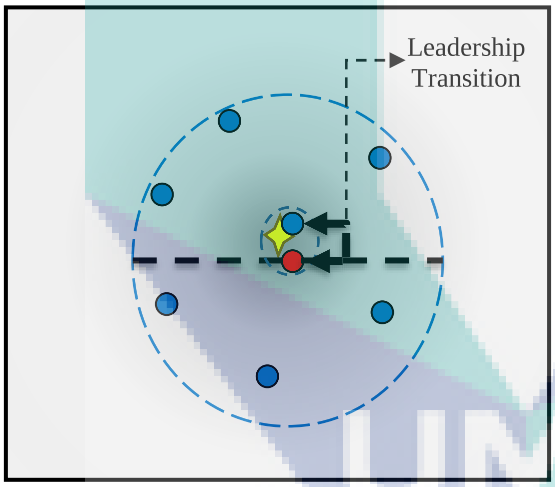


Figure 4.10A Leadership Transition

Figure4.10B Op.4 Leadership Transition



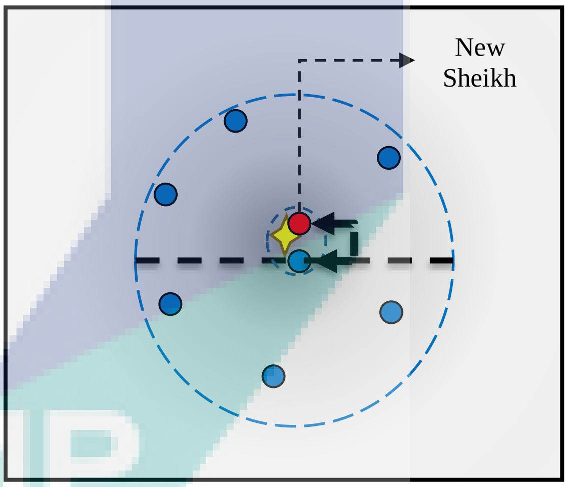


Figure4.10C Semi-Circular Distribution

Figure4.10D 4 Leadership Transition

Figure 4.10 Exploitation Analysis

In Equation (4.4), there are three main variables or parameters which controls the distribution of the families around the Sheikh's tent. These variables are the position of the Sheikh's tent, the angle, and the random distance (radius) between the families and the Sheikh's tent. Note that each of these variables has its own effect on the distribution of the families (i.e. solutions). From equation (4.4), the new location X_c equals the Sheikh's (σ_c) location multiplied by the square root of the random radius $(Rand)$ and by $cos(\theta)$. As $cos()$ is within $[-1,1]$, and $Rand$ is within $[0,1]$, four main observations are evident:

1. The position of the normal families is always *equal* to the origin of the coordinate system or the Sheikh's tent when $Rand$ is *equal* to 1 and θ is *equal* to 0 or 2π, as follows:

$$X_c = \sigma_c * \sqrt{1} * \cos(2\pi) \rightarrow X_c = \sigma_c * 1 * 1 \rightarrow X_c = \sigma_c$$

$$X_c = \sigma_c * \sqrt{1} * \cos(0) \rightarrow X_c = \sigma_c * 1 * 1 \rightarrow X_c = \sigma_c$$

2. The position of the normal families is always *close* to the origin of the coordinate system or the Sheikh's tent when $Rand$ is *close* to 1 and θ is *close* to 0 or 2π.

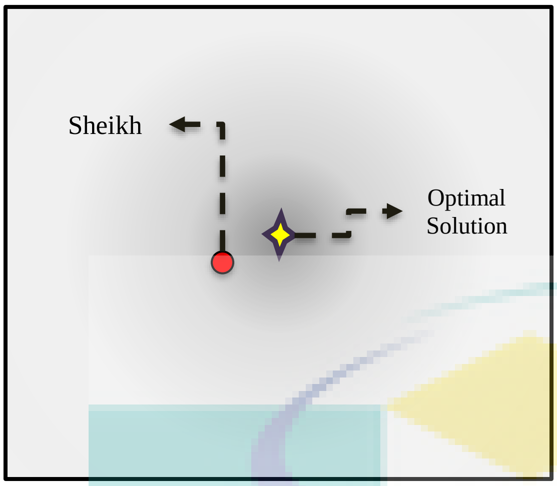
3. The distance between the normal families and the Sheikh's position *increases* as the value of $Rand$ *decreases* or as the value of θ is close to π.

4. It is nearly impossible to give the same exact random values for both variables due to each the difference in the range of each variable ($Rand \in [0,1], \theta = [0,2\pi]$). Whenever $Rand$ and θ produce the same values, the local search is normally performed because each one is used in a different mathematical operator, square root and cos function, Thus, two different values will be generated.

From these observations, the semi-circular distribution has a great impact on the exploitation part in NPO, especially when the value of $Rand$ is close to 1. However, this step, in some cases, can help the NPO algorithm to escape from local minima when the value of $Rand$ is small enough because the normal families will be far away from the center or the Sheikh's tent.

4.4.2 Exploration

The primary meeting and the families searching step represent the exploration ability of the algorithm. The first step represents the initialization stage, in other words, it is responsible for initializing the first solutions or the position of each Sheik in the search space. The second step represents the global search ability of the algorithm, in other words, it is implemented when there is no family with a better fitness than that of the current Sheikh. This step helps the families to explore the search space by moving towards different directions using levy flight function. Figure 4.13 (A-H) simulates the exploration process of NPO.



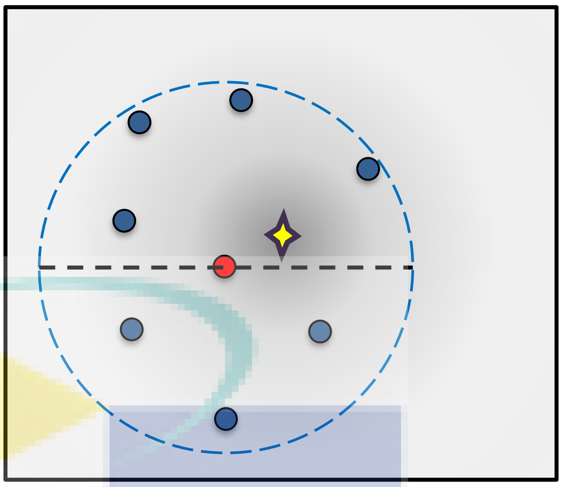


Figure 4.11A Initialization (One Sheikh) Figure 4.11B Semi-circular distribution

Figure 4.11 Illustration of the first two steps of NPO

In the Figure 4.11A and Figure 4.11B, the leader is initialized and established his clan in a semi-circular shape. The position of the leader is closer than all other families to the optimal solution, meaning that there is no family in the clan has a better positon than their leader. This leads to execute the third step (i.e., families searching) or the exploration.

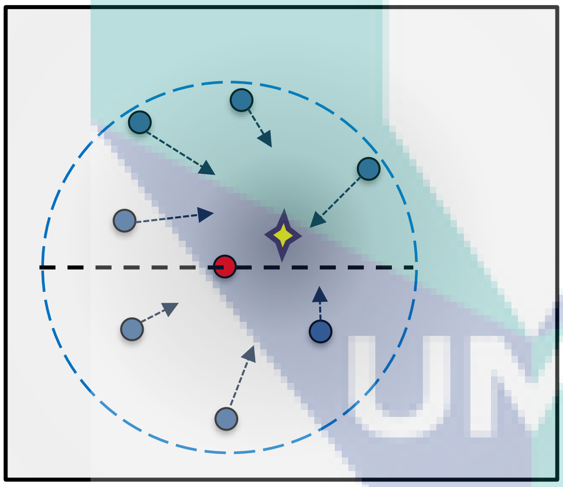
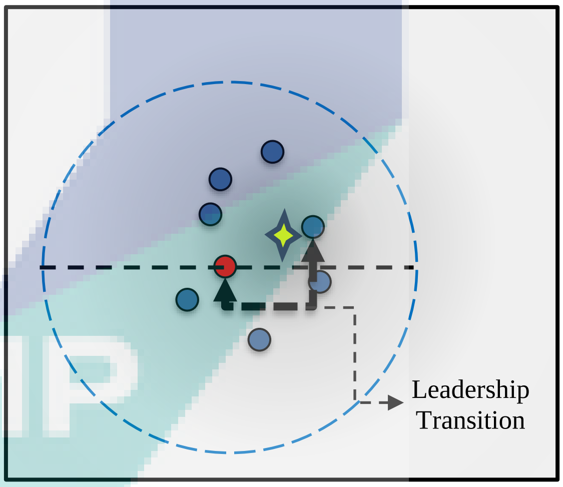



Figure 4.12A Families Searching Figure 4.12B Leadership Transition

Figure 4.12 Illustration of the exploration and exploitation of NPO

Based on equation 4.5, all families tried to move towards the current best solution which is the leader. Once there is a family found a new better position – or near to the optimal solution – then it will be the new current local best (i.e., new leader) via executing the fourth step (Leadership Transition).

It is worth to mention that this simulation illustrates the searching process of only single clan – or swarm – while in the NPO, there are C clans are exploring the search space at the same time. Meaning that, the local search is performed multiple times in different places in the search space, and there are several local best solution for each clan.

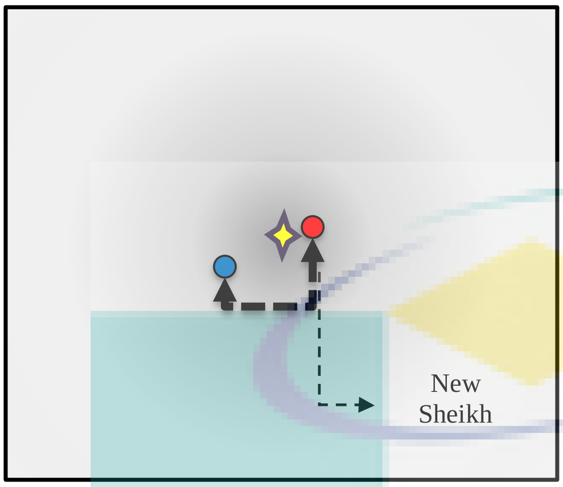


Figure 4.13A Leadership Transition

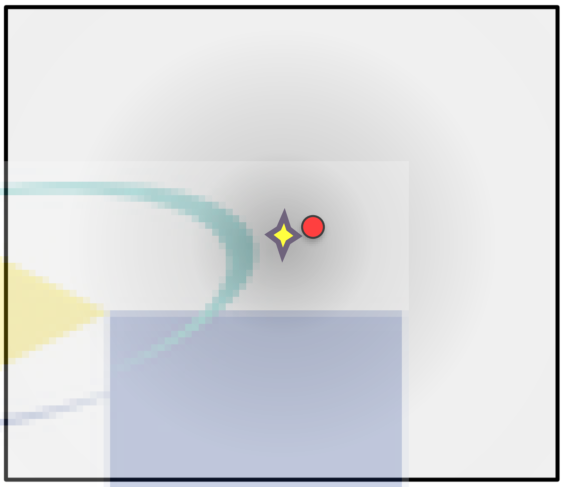


Figure 4.13B Leadership Transition

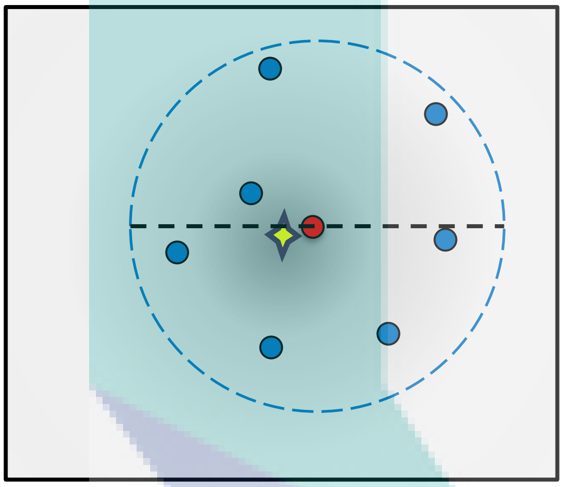


Figure 4.13C Semi-circular distribution

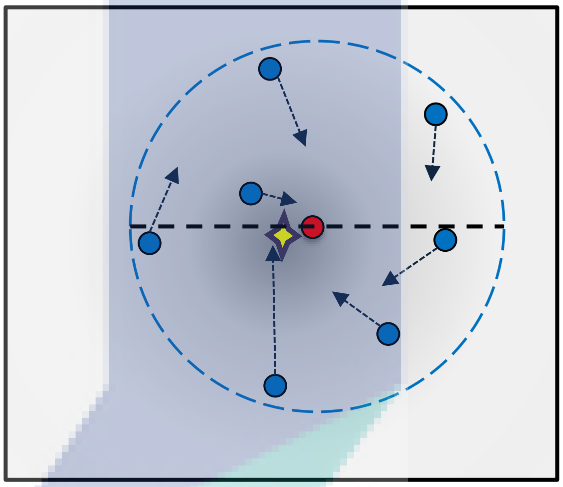


Figure 4.13D Leadership Transition

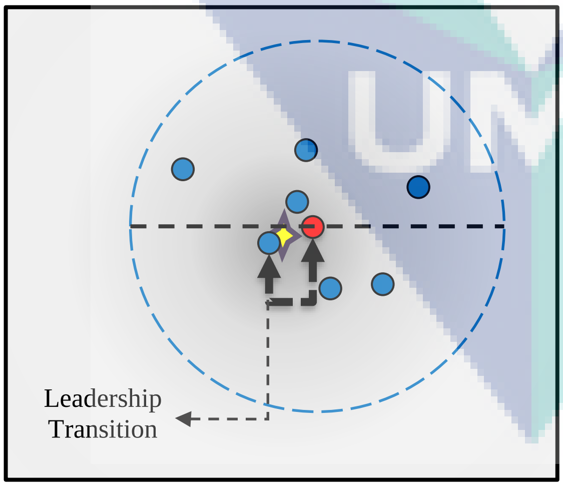


Figure 4.13E Leadership Transition

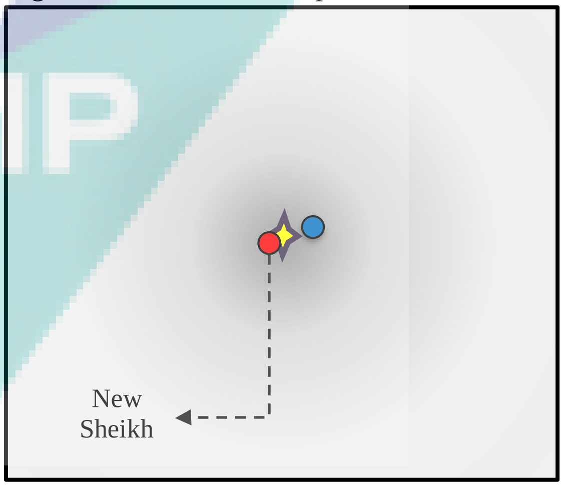


Figure 4.13 Leadership Transition

Figure 4.13 Exploration Ability of NPO

The families searching step takes charge as a global search mechanism. This step is used when there is no better family in the clan than the Sheikh's family, meaning that they are positioned in places with limited food sources. Therefore, the families must find better positions to have a chance of taking the leadership position from the current Sheikh. Based on Equation (4.5), there are two main components, the current position, and the step size factor. The step size factor consists of two main factors as well, the clan area (a_c) and the Levy flight equation. The clan area is an average value of the distances between all the families and their Sheikh. It determines the quality of all the families or solutions in the clan. In other words, a_c is an indicator of how good the families are distributed in the clan (i.e. search space); if a_c is big, the average distance between the Sheikh and the families is big and this results in a good exploration ability.

4.4.3 Graphical Illustration of Balancing Mechanism

As earlier stated, the last step in the NPO (the periodical meeting) represents a balancing mechanism between exploration and exploitation. This step is generally called the "Meeting Room Approach" or MRA. It is a multi-swarm cooperative approach which involves only the Sheikhs. In other words, the Sheikhs occasionally meets and share their positional information. Specifically, the best Sheikh among the Sheiks provides the information for the other Sheiks to update their positions. Figure 4.14 (A-G) simulate and illustrate the effect of MRA on the NPO algorithm.

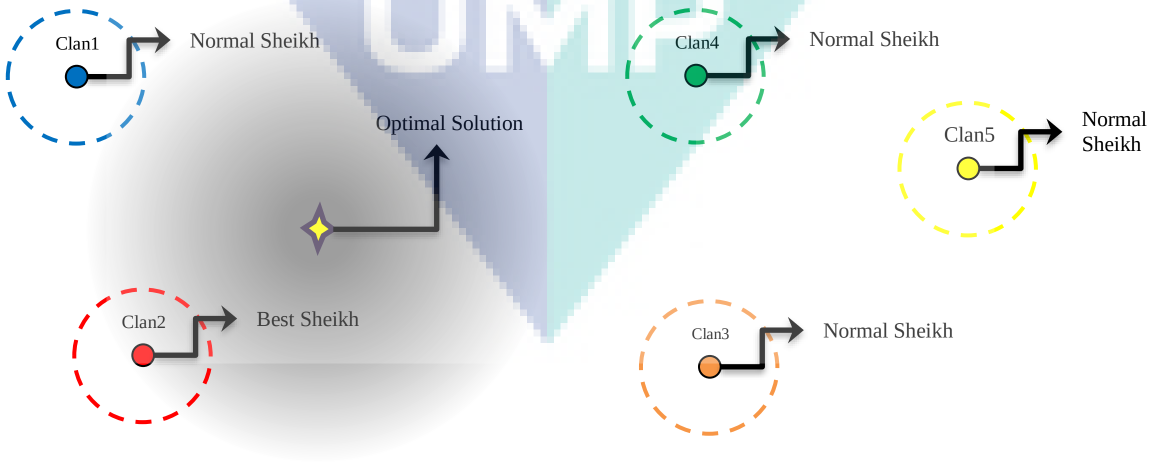


Figure 4.14A The best sheikh has been determined

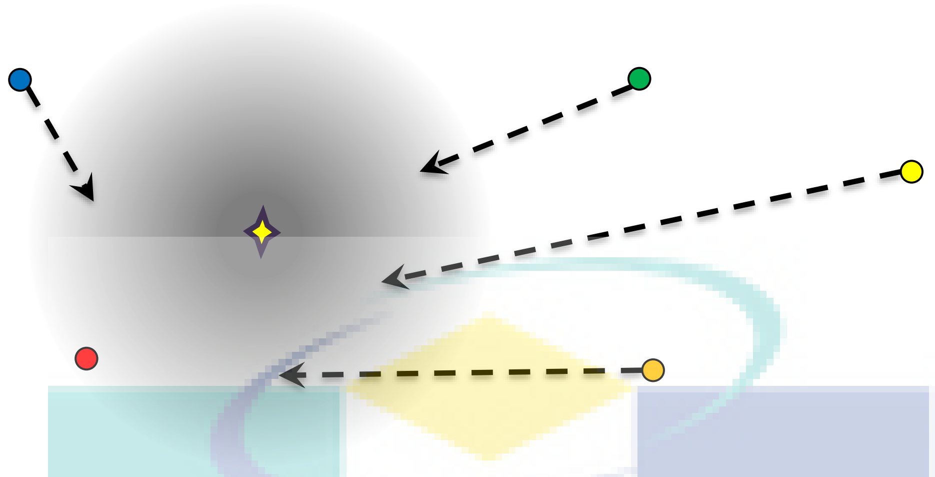


Figure 4.14B The normal sheikhs move toward the best sheikh

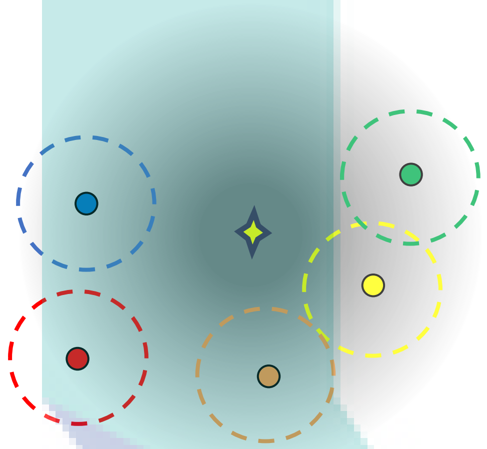


Figure 4.14C The sheikhs established their clans in the new positions using OP.2

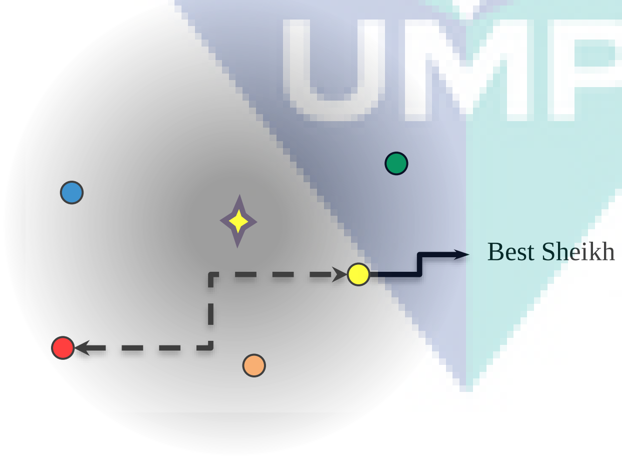


Figure 4.14D New Best Sheikh has been determined

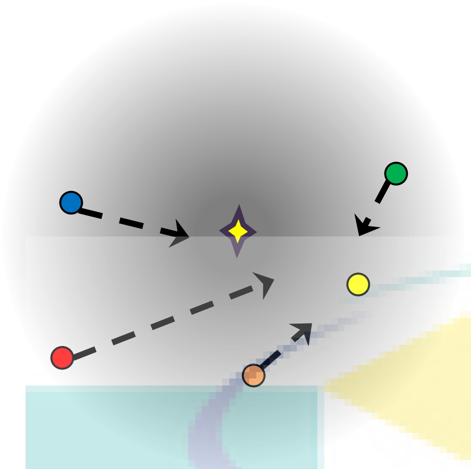


Figure 4.14E The normal sheikhs move toward the best sheikh

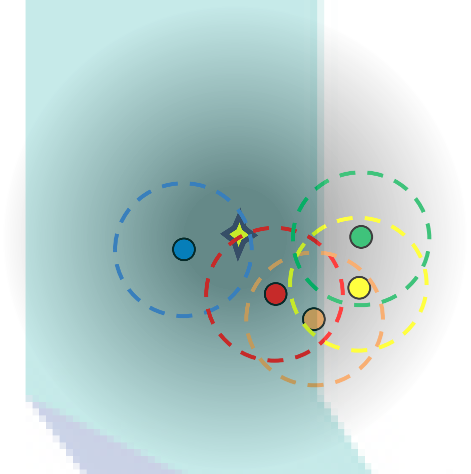


Figure 4.14F The sheikhs established their clans in the new positions using OP.2

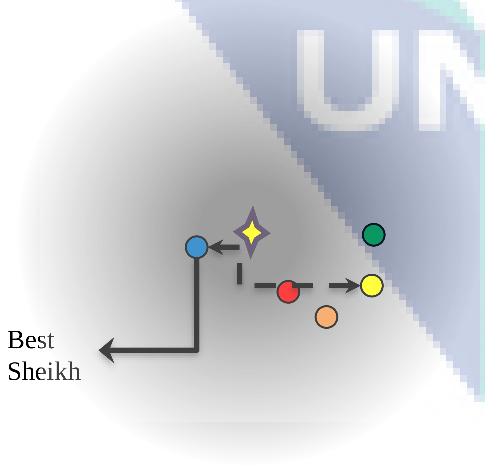


Figure 4.14G New Best Sheikh has been determined

Figure 4.14 MRA Balancing Mechanisim

It is obvious from the figures that all the swarms or clans search for the optimal solution in two different ways; the first way is when only their Sheikhs move towards the best Sheikh (representing another exploration ability of NPO), while the second way is when each swarm, including their Sheikhs and families, are moving together around the best Sheik, meaning that the swarms are exploiting the search space (another form of exploitation ability). This behavior is due to the effect of the direction variable in Equation (3.8). The direction variable (Ψ) guides the Sheikhs towards better places; however, the value of ΔPos is normalized and helps the Sheikhs to find better places rather than the place of the best Sheikhs. The simulation presented above focused only on the effect of MRA on the searching process of NPO. However, the interaction between all steps on the performance of the proposed NPO algorithm is presented Figure 4.15 below. Which illustrate the effect of MRA, semi-circular distribution, and leadership transition on the search process of NPO.

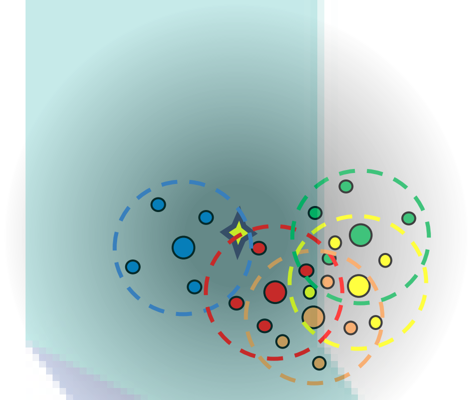


Figure 4.15A All swarms and families distributed using step 2

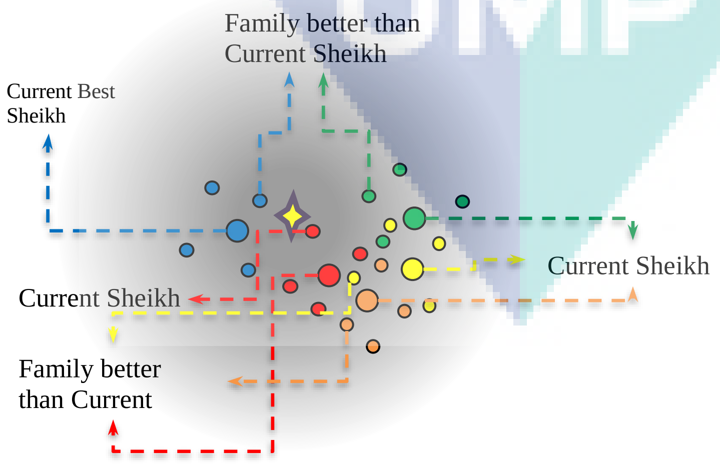


Figure 4.15B Determin the new Sheikhs for all clans using the explotation step

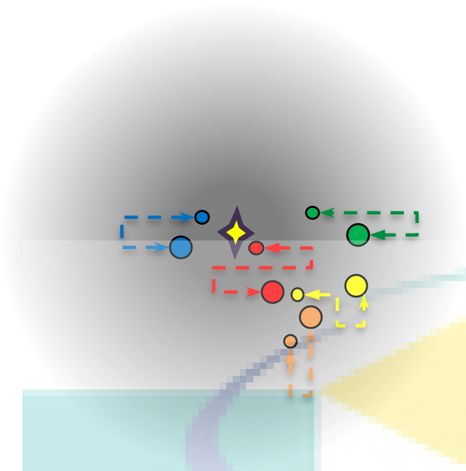


Figure 4.15C Leadership Transition for all clans

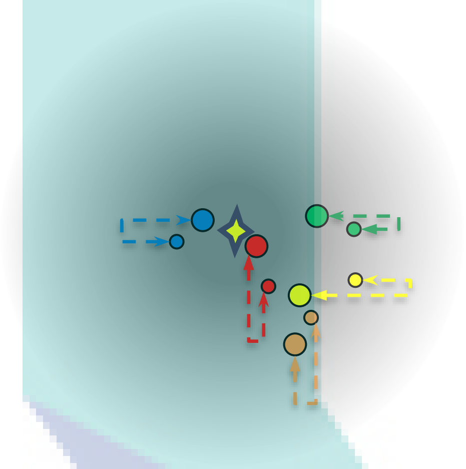


Figure 4.15D New Sheikhs have been determined

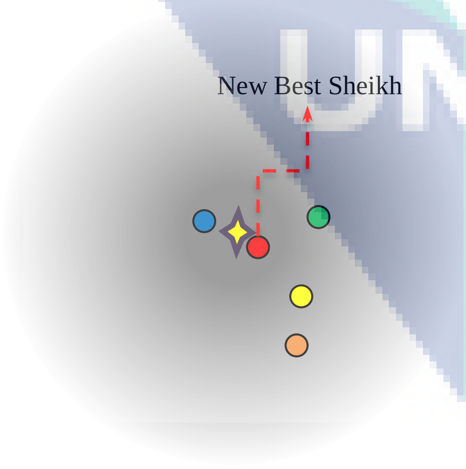


Figure 4.15E New best Sheikh has been determined for MRA

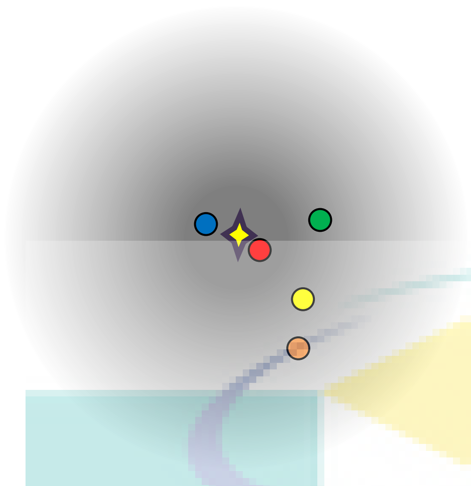


Figure 4.15F All Sheikhs establesh their new clans using step 2
Figure 4.15 Simulation of all NPO operators

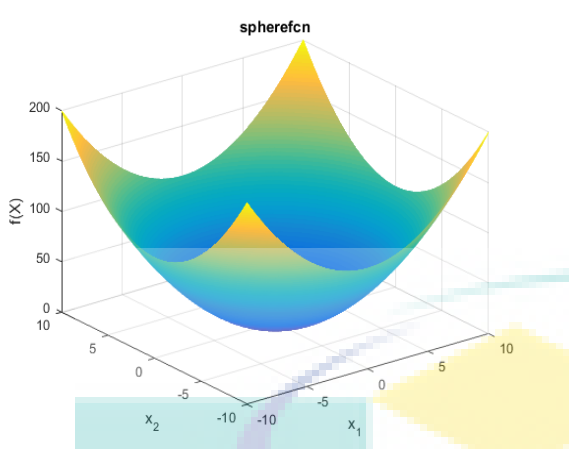
The interaction between the clans is very clear in the figures above, the figure showed how fast the searching process can be. Because the normal families can be the best Sheikh ever once they find good positions, which leads to good balancing between the exploration and the exploitation capability of NPO which is the first contribution of the thesis.

4.5 Example: NPO for Solving Sphere Problem

Sphere is a very common benchmark test function, it is continuous, convex and unimodal, where there is only a single local minimum solution. The plot of two dimensions and contour of the function Sphere function is given in Figure 4.16, it can be formulated as follows:

$$f(x) = \sum_{i=1}^{D} x^2 \qquad 4.11$$

Where x represents the vector of decision variables, while D represents number of dimensions. In this section, NPO is applied to solve the sphere function. The settings of this example with the values of the NPO parameters are given in the following table:



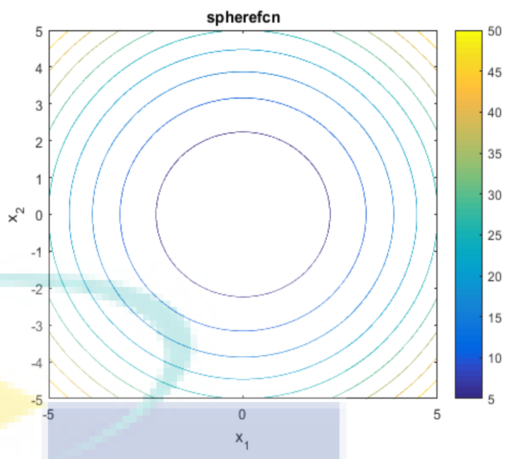


Figure 4.16A The plot of two dimensions Figure 4.16B The contour of the function

Figure 4.16 Sphere Test Function

Table 4.2 Parameter Settings for the example

Parameter	Value
UpperBound (UB) , LowerBound (LB)	100, -100
No. of Dimensions (D)	2
No. of Clans $(\#C)$	2
No. of Families $(\#F)$	3
No. of Iterations $(\#T)$	2

Sphere test function is a unimodal test functions, which needs for the exploitation steps (i.e., semi-circular distribution and leadership transition). The sequence of the solution is given as follows:

Note: All numbers in the following steps are generated randomly, except for the fitness values, they have been calculated based on the objective function.

(STEP 1) Read the initial values for $\#C$, $\#F$,$\#T$.

(STEP 4) Define the objective function, which is: what are the best decision variables (x) which manimize the objective function $f(x)$, as follows:

$$\min f(x) = \sum_{i=1}^{D} x_i^2 \ where\ x_i \in [UB, LB], \text{for all } i = 1,2, \dots D.$$

The optimal value for this test function $f(x^*) = 0$ at x^* = (0,0, … , 0).

(STEP 5) For each clan, initialize the positions of their Leaders σ_c^O via equation 4.1, as follows:

$\sigma_1^O.Position(1)$ = 100 – (-100) × 0.80725 + (-100) = 61.4498

$\sigma_1^O.Position(2)$ = 100 – (-100) × 0.00933 + (-100) = -98.1336

$\sigma_2^O.Position(1)$ = 100 – (-100) × 0.48635 + (-100) = -2.7287

$\sigma_2^O.Position(2)$ = 100 – (-100) × 0.53975 + (-100) = 7.9504

(STEP 6) Calculate the fitness value for the leaders using the identified $f(x)$ in step 2, as follows:

$$\sigma_1^O.Fitness = 13406.288$$

$$\sigma_2^O.Fitness = 70.6565$$

(STEP 7) Start the main iteration Itr from 1 until $\#T$

(STEP 8) For c = 1 to $\#Clan$

(STEP 9) Distribute the families in the clan c around σ_c^O via equation 4.4, as follows:

When $c = 1$:

$$x_{11} = \sigma_1^O$$

$x_{12}.Position(1) = 61.4498 \times \sqrt{0.44} \times cos(3.88)$ = -30.1445

$x_{12}.Position(2) = -98.1336 \times \sqrt{0.67} \times cos(0.94) = -47.375202$

$x_{13}.Position(1) = 61.4498 \times \sqrt{0.51} \times cos(4.31)$ =-17.18572

$x_{13}.Position(2) = -98.1336 \times \sqrt{0.83} \times cos(6.03)$ = -86.55371

Where $c = 2$:

$x_{22}.Position(1) = -2.7287 \times \sqrt{0.11} \times cos(4.21) = 0.435779$

$x_{22}.Position(2)$= 7.9504 × $\sqrt{0.27} \times cos(1.18)$ = 1.573657

$x_{23}.Position(1) = -2.7287 \times \sqrt{0.95} \times cos(5.91)$ =-2.47654

$x_{23}.Position(2) = 7.9504 \times \sqrt{0.55} \times cos(2.09)$ = -2.92561

(STEP 10) Calculate the fitness value for each family in the clans, as follows:

Where $c = 1$:

$$x_{12}.Fitness = 3153.10006$$

$$x_{13}.Fitness = 7786.8936$$

Where $c = 2$:

$$x_{22}.Fitness = 2.66629$$

$$x_{23}.Fitness = 14.69244$$

(STEP 11) If $x_{1i} < \sigma_1^O$ then swap them.

Where $c = 1$:

If there is an (x_{1i}) is better (min) than the leader σ_1^O, then $\sigma_1^B = x_{1i}$.

Where $c = 2$:

If there is an (x_{2i}) is better (min) than the leader σ_2^O, then $\sigma_2^B = x_{1i}$.

(STEP 12) If σ_c^B is better (min) than σ_c^O then switch them.

Where $c = 1$:

$\sigma_1^B < \sigma_1^O$ then , $\sigma_1^O = \sigma_1^B$.

Where $c = 2$:

$\sigma_2^B < \sigma_2^O$ then , $\sigma_2^O = \sigma_2^B$.

Note: The steps (STEP 13) To (STEP 18) will not be executed, because this is the ELSE part.

(STEP 19) If $\leq \#Clans$, then Go To (STEP 8), Otherwise Go To (STEP 20).

(STEP 20) Determine the best leader, then the leader of the second clan is the best, as follows: $\sigma^E = \sigma_2^O$ = 2.66629.

(STEP 21) The value of direction variable $\Psi = 1$, based on equation 4.9.

(STEP 22) Calculate ΔPos via equation 4.8, as follows:

$$\Delta Pos = 1 \times \left(\frac{\sqrt{(0.435779 - -30.1445)^2 + (1.5736 - -47.3752)^2}}{2} \right) = 28.8580$$

(STEP 23) For each normal leader, update their positions, as follows:

$\sigma_1^{new} = [-30.1445, -47.3752] + 28.8580 \times ([0.435779, 1.5736] - [-30.1445, -47.3752] =$ [852.3412, 1365.24]. Meaning that:

$$\sigma_1^{new}.Position(1) = 852.3412$$

$$\sigma_1^{new}.Position(2) = 1365.24$$

(STEP 24) Calculate the fitness value for the new normal leader, as follows:

$$\sigma_1^{new}.Finess = 4917421.548$$

(STEP 25 – STEP 26) If σ_1^{new} is better (min) than the σ_1^N then swap them. Otherwise, keep the original value. It can be noted that the fitness value of the new normal leader is worst (higher) than the original normal leader, then, it will keep the original positions.

(STEP 27) If the is no more normal leaders, then Go To (STEP 28), otherwise Go To (STEP 23).

(STEP 28) If $Itr > \#T$ then Go To (STEP 29), Otherwise Go To (STEP 7).

(STEP 29) Return the best solution $\sigma^E = 2.66629$.

The steps above showed in details the main sequential of the process of the algorithm when it tried to solve the Sphere test function. It is obvious that the algorithm did not execute the exploration part (Steps 13 – 18), due to the nature of the Sphere function which is a unimodal function, meaning that it requires exploitation more than exploration. Only single iteration has been occurred during the previous example, more iteration will attain better results the results. This function will be tested in details in next chapter.

4.6 Summary

This chapter explained in details the proposed metaheuristic, which is called 'Nomadic People Optimizer', or NPO. The chapter started by illustrating the source of inspiration of nomad people, presenting in details the life of bedwins, and the inspired laws and observations. The mathematical model and the five steps are given in this chapter as well. Finally, the exploration and the exploitation capability of NPO with the proposed balancing mechanism – called Meeting Room Approach 'MRA' – have been analysed in details. The results of NPO over unconstrained benchmark test function are going to be presented and discussed in the next chapter. While the evaluation of the proposed MRA are going to be presented in the following chapter over the Particle Swarm Optimization (PSO) algorithm.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

In the previous chapter, the source of inspiration for the proposed NPO metaheuristic has been described. Additionally, the mathematical model and the main pseudocode were given. As mentioned, NPO consists of five steps, each step has a specific need and purpose. The last step, which is the periodical meeting, represents the proposed meeting room approach, which is the first objective of this thesis. It helps NPO to balance between the exploration and the exploitation ability, by guiding the best sheikhs toward better positions in the search space.

To benchmark the performance of NPO metaheuristic, the current chapter presents a complete evaluation for NPO with the necessary statistical analysis. Overall, this chapter is divided into five main sections. The first sections presents the experimental settings, which consists of two subsections, the first subsections illustrates the new combinations test functions, while the second subsections illustrates the simulation settings. The second section presents the results of NPO over the 60 test functions. In addition, the section also presents the results of NPO over the large-scale problems, which are 18 test functions selected form the 60 test functions mentioned above with different number of dimensions. In third section, the convergence curves are displayed. The fourth section presents the analysis of the exploration and the exploitation ability of NPO, while the fifth section discusses the attained results, and presents a complexity analysis for NPO. Finally, the last section gives the summary.

5.2 Comparison and Simulation Settings

All the experiments, including NPO and the previously mentioned 60 benchmark test functions were executed on a personal computer (Core i7, 3.60 GHz, 16 GB of RAM, 64-bit Windows 10 operating system) using MATLAB 2018a .The performance of NPO was compared to that of six well-known metaheuristics (Particle Swarm Optimization (PSO2011), Artificial Bees Colony (ABC), Flower Pollination Algorithm (FPA), Grey Wolf Optimizer (GWO), and Firefly Algorithm (FFA)). The experiments were executed in 30 different runs, and the best, worst, median, mean, and standard deviation were recorded. Table 5.1 showed the specific/default parameters for the metaheuristics mentioned above. Appendix B contains more details about the metaheuristics.

NPO was compared to the other metaheuristics based on the mentioned statistical parameters based on 30 run times. Also, NPO is compared to the other metaheuristics based on a statistical tests which is the Mean Absolute Error (MAE), and the Wilcoxon Signed-Rank test. To establish the speed of the NPO in converging to the optimal solution, a convergence analysis for all the algorithms was performed. The results of the 30 runtimes (means and standard deviation) are compared to those of the mentioned metaheuristics.

Table 5.1 The specific paramaters used in the studies metaheuristics

Algorithm	Parameter	Settings
PSO2011	Swarm Size S.S	50
	Inertia weight ω	Linearly Decrease (0.9 – 0.1)
	cognitive parameter $c1$	1.49
	social parameter $c2$	1.49
ABC	Colony Size C.S	50
	No. Food Source	C.S / 2
	Limit	50
FPA	Population Size P.S	50
	Switch Probability P	0.8
	Levy flight λ	1.5
GWO	Swarm Size S.S	50
	a	Linearly Decrease (2 – 0.1)
FFA	Swarm Size S.S	50
	a	0.5
	β_{min}	0.2
	γ	1.0
	δ	0.96
NPO	Swarm Size ($\sigma \times \#F$)	50 (5×10)

5.3 Results

This section presents the results of proposed NPO, it is divided into two subsections. In the first subsection, the performance of NPO over the unconstrained test function is presented, while the second subsections presents the performance of NPO over the large-scale problems.

5.3.1 NPO for unconstrained test functions

After executing and recording all the experiments over the 60 benchmark test functions, the outcomes showed that the NPO exerted superior performance and could reach the optimal solution for many test functions. Although some of these test functions had been exceptionally challenging to solve and their best results could not be efficiently arrived at with the NPO, the algorithm was able to reach values very close to their ideal best results. Figure 5.1 (A-F) displays several selected test functions. Table 5.2 presents the results of NPO and the other five metaheuristics over the 60 test functions.

All algorithms were evaluated 30 run times on 60 test functions, which means each algorithm was ran for a total of 1800 times. Figure 5.2 illustrates that the NPO arrived at the best solution for 52 tests, out of 60, while GWO ranked second with 43 best solutions. ON the other hand, ABC and FPA managed to solve 42 tests each, FA with 34 tests, and finally, PSO solved only 21 tests. Figure 5.2, Figure 5.3, and Figure 5.4 portray the complete results of all tests, the results of unimodal only, and the results of multimodal results, respectively.

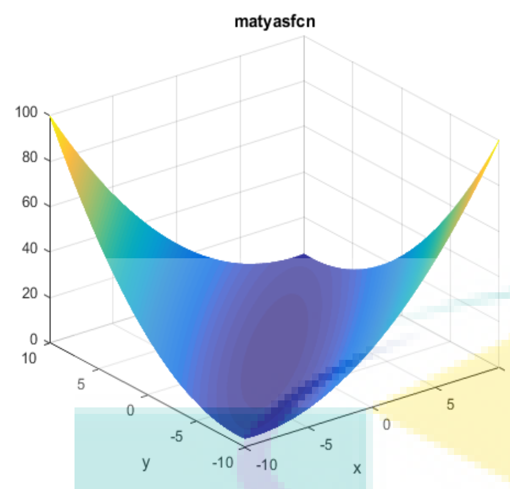


Figure 5.1A Matyas test function

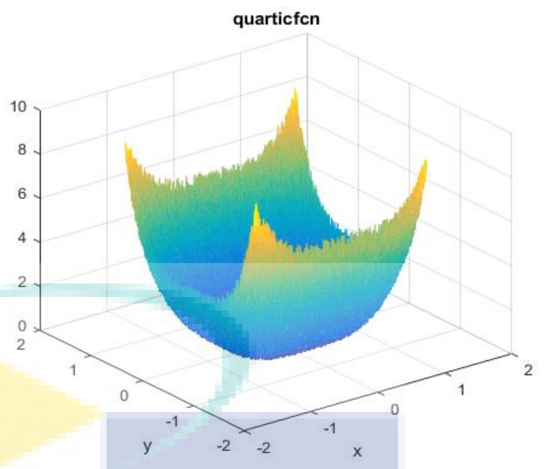


Figure 5.1B Quartic test function

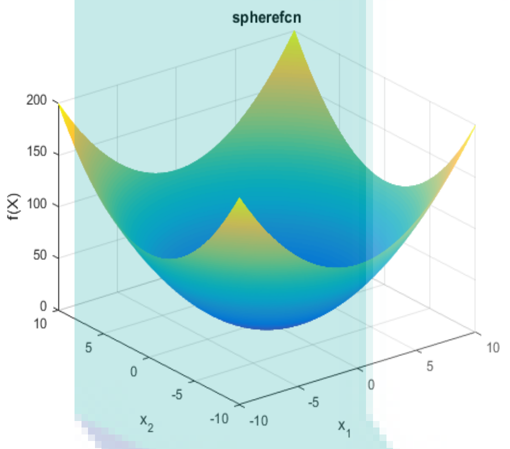


Figure 5.1C Sphere test function

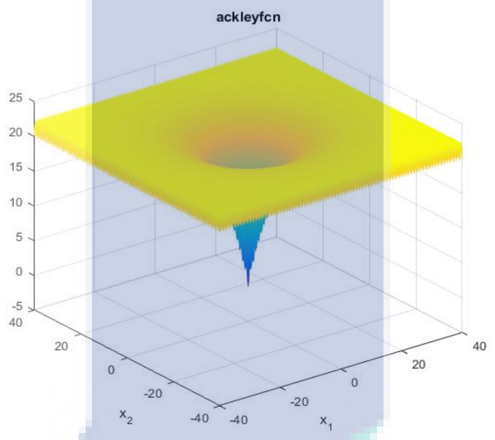


Figure 5.1D Ackely test function

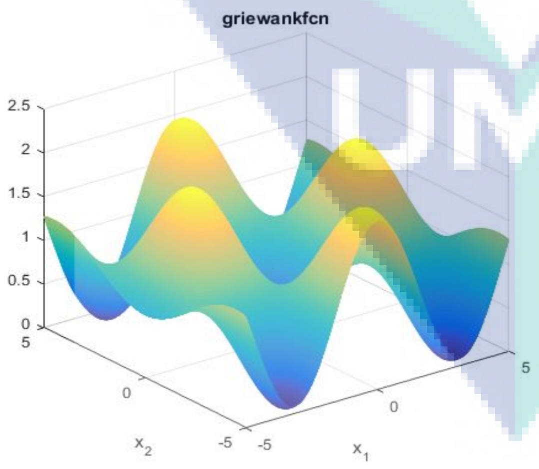


Figure 5.1E Griewank test function

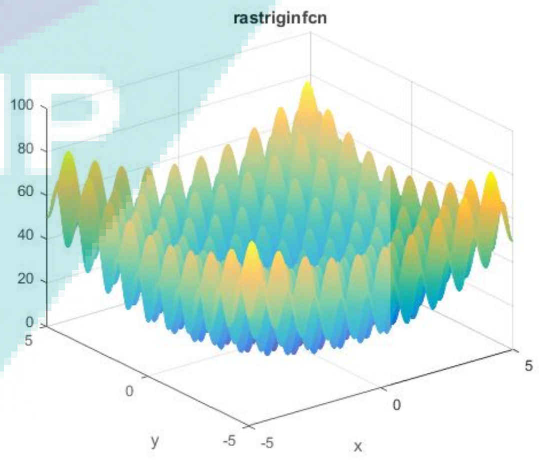


Figure 5.1F Rastrigin test function

Figure 5.1 3D illustration of some benchmark test functions

Table 5.2 Results of the metaheuristics over benchmark test functions

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
	Best	-199.9877	-200	-200	-200	-200	-200
	Worst	-199.8449	-200	-200	-200	-199.9994	-200
1	Median	-199.9545	-200	-200	-200	-199.9997	-200
	Mean	-199.9439	-200	-200	-200	-199.9997	-200
	S.D	0.037191	0	0	5.49E-10	0.00013675	0
	Best	2.7457E-06	1.55E-12	0	5.56E-10	2.56E-12	1.20E-07
	Worst	0.0014077	1.31E-08	0	0.76207	1.83E-09	0.000948752
2	Median	0.000090535	2.45E-10	0	2.48E-08	2.35E-10	2.64811E-05
	Mean	0.00020231	1.15E-09	0	0.025402	4.04E-10	0.000104974
	S.D	0.00028704	2.63E-09	0	0.13913	4.78E-10	0.0033127
	Best	25.7848	1.14E-07	1.0499	1.14E-63	1.75E-06	0
	Worst	156.9656	6.2875E-04	15.015	6.2875E-60	9.25E-06	0
3	Median	83.8289	2.83E-62	6.5799	2.83E-62	4.09E-06	0
	Mean	85.1968	2.68E-60	7.1191	2.68E-60	4.11E-06	0
	S.D	34.7038	1.18E-59	3.8435	1.18E-59	1.67E-06	0.00E+00
	Best	-0.99974	-1	-1	-1	-1	-1
	Worst	-0.95692	-0.998047	-1	-1	0.00E+00	-1
4	Median	-0.99655	-1	-1	-1	-1.00E+00	-1
	Mean	-0.99273	-0.999917	-1	-1	-7.33E-01	-1
	S.D	0.009772	3.59E-04	0.00E+00	1.51E-07	4.50E-01	0.00E+00
	Best	1.18E-04	3.61E-05	0	6.25E-10	3.84E-12	2.37E-07
	Worst	2.60E-02	0.068356283	0	1.66E-06	4.68E-10	0.89713
5	Median	2.15E-03	0.001941576	0	1.59E-07	9.86E-11	0.32655
	Mean	5.13E-03	0.006420366	0	3.32E-07	1.22E-10	0.34166
	S.D	7.08E-03	0.012799033	0	4.09E-07	1.05E-10	0.23695
	Best	3.5745E-08	4.31E-15	9.96E-57	4.94E-247	2.45E-12	0
	Worst	0.000055609	1.36E-11	5.91E-47	9.01E-210	4.68E-10	0
6	Median	7.4128E-06	1.36E-12	1.36E-50	7.65E-222	7.97E-11	0
	Mean	0.000013438	2.57E-12	3.14E-48	3.02E-211	1.21E-10	0
	S.D	0.000014366	3.66E-12	1.11E-47	0	1.22E-10	0
	Best	1.44E-01	3.77E-05	0.43335	4.43E-13	4.17E-09	0
	Worst	1.76E+01	7.45E-03	14.5569	1.66E-08	2.96E-06	3.94E-39
7	Median	3.56E+00	1.53E-03	3.5848	1.64E-09	1.90E-07	3.89E-153
	Mean	4.48E+00	2.13E-03	4.4113	3.01E-09	4.49E-07	3.50E-20
	S.D	4.07E+00	1.98E-03	3.411	3.82E-09	6.53E-07	1.31E-40
	Best	3.75E-07	0	0	0	3.05E-12	0
	Worst	6.14E-04	0	0	0	2.50E-03	0
8	Median	1.70E-05	0	0	0	5.30E-11	0
	Mean	4.80E-05	0	0	0	8.32E-05	0
	S.D	1.13E-04	0	0	0	4.56E-04	0
	Best	1.55E-07	0.00E+00	0	0	1.14E-11	0
	Worst	5.48E-03	1.93E-07	0	0	4.45E-10	0
9	Median	2.04E-04	1.72E-10	0	0	8.22E-11	0
	Mean	8.37E-04	1.16E-08	0	0	1.08E-10	0
	S.D	1.53E-03	3.68E-08	0	0	9.46E-11	0
	Best	0.0015737	0.001566855	0.0015669	0.001566855	0.0015669	0.0015669
	Worst	0.0094556	0.001587719	0.0015672	0.001567308	0.0015671	0.0015669
10	Median	0.0024329	0.001567421	0.0015669	0.001566888	0.0015669	0.0015669
	Mean	0.0035316	0.00156986	0.0015669	0.001566911	0.0015669	0.0015669
	S.D	0.002101	4.71E-06	8.60E-08	8.83E-08	4.02E-08	1.36E-08
	Best	0.29259	0.292578645	0.29258	0.29258	0.29258	0.29258
	Worst	0.29659	0.292589884	0.29258	0.29258	0.29661	0.29258
11	Median	0.29391	0.292579232	0.29258	0.29258	0.29258	0.29258
	Mean	0.294	0.292580317	0.29258	0.29258	0.29288	0.29258
	S.D	0.0012469	2.50E-06	2.35E-07	1.72E-07	8.84E-04	1.39E-15
	Best	-47.1355	-50	-50	-50	-50	-50
	Worst	24.9203	-50	-50	-49.9997	-50	-50
12	Median	-27.1536	-50	-50	-49.9999	-50	-50
	Mean	-24.2545	-50	-50	-49.9999	-50	-50
	S.D	1.69E+01	2.96E-14	3.54E-14	5.29E-05	1.52E-06	0
	Best	-209.8977	-210	-210	-209.99	-209.9999	-210
	Worst	-159.3806	-210	-210	-74.693	-209.998	-209.98
13	Median	-204.2893	-210	-210	-209.995	-209.9996	-210
	Mean	-201.6952	-210	-210	-188.397	-209.9994	-209.995
	S.D	1.44802	6.20E-13	4.92E-07	38.2347	5.42E-04	6.1259

Table 5.2 Results of the metaheuristics over benchmark test functions

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
	Best	246.5546	1.89E+02	17.2851	4.40E-29	4.6417	0
	Worst	520.3061	3.16E+02	169.1789	6.74E-26	31.1956	0
14	Median	402.8341	2.51E+02	44.2972	1.65E-29	18.4768	0
	Mean	389.7976	2.48E+02	53.1959	6.38E-27	17.9621	0
	S.D	72.833	3.47E+01	31.1372	1.39E-26	6.80E+00	0
	Best	0.024323	3.25E-16	2.21E-12	0.00E+00	1.44E-08	0
	Worst	0.2799	4.41E-13	1.48E-06	0.00E+00	7.69E-07	0
15	Median	0.087994	1.19E-14	2.20E-10	0.00E+00	1.07E-07	0
	Mean	0.11365	4.47E-14	7.03E-08	0.00E+00	1.43E-07	0
	S.D	0.071194	8.75E-14	2.81E-07	0	1.41E-07	0
	Best	1.3389	0.1153898	0.028022	1.85E-04	0.00409	6.00E-08
	Worst	7.7498	0.294229888	0.56196	0.001033197	0.096157	8.24E-05
16	Median	5.4099	0.206504048	0.13346	4.47E-04	0.015618	9.58E-06
	Mean	6.9606	0.195930447	0.17695	4.47E-04	0.025424	1.74E-05
	S.D	0.6477	0.055478056	0.12362	2.11E-04	0.023129	2.04E-05
	Best	6.412	1.73E-14	24.4649	1.84E-41	0.20607	0
	Worst	15.3984	1.38E-13	96.0046	2.27E-39	0.49932	0
17	Median	12.912	4.33E-14	52.2212	3.85E-40	0.33775	0
	Mean	8.3662	4.92E-14	52.7072	5.61E-40	0.34615	0
	S.D	0.4647	2.84E-14	16.3115	6.14E-40	0.068941	0
	Best	57.2433	30.76537235	10.4292	2.71E-19	0.038303	0
	Worst	80.7359	52.86893997	21.4516	4.46E-17	0.094114	0
18	Median	68.9134	40.23089612	15.5261	7.06E-18	0.061089	0
	Mean	68.8236	40.79692823	15.7964	1.18E-17	0.063286	0
	S.D	4.6243	6.529544516	3.1328	1.34E-17	0.0157	0
	Best	1.7985	1.30E-14	47493.2929	8.28E-41	0.27895	0
	Worst	4.3046	1.02E-13	5.70E+19	4.06E-39	50.1202	0
19	Median	3.1041	4.37E-14	2392815161	4.53E-40	0.36643	0
	Mean	2.5986	4.52E-14	1.90E+18	9.00E-40	3.2888	0
	S.D	1.0657	2.25E-14	1.04E+19	1.02E-39	9.9447	0
	Best	9.5101	2.35E+00	0.44804	4.34E-107	3.90E-23	0
	Worst	19.1102	7.85E+00	2185.4865	6.25E-91	1.09E-17	0
20	Median	12.0145	4.41E+00	26.123	4.31E-98	8.48E-20	0
	Mean	13.4548	3.98E+00	269.1776	2.35E-92	7.55E-19	0
	S.D	3.0017	1.09E+00	575.2657	1.14E-91	2.02E-18	0
	Best	1.2945	0	2.66E-52	0	0.0012864	0
	Worst	5.1309	0	2.34E-39	0	0.0053216	0
21	Median	3.8818	0	6.97E-46	0	0.0027475	0
	Mean	2.7707	0	8.28E-41	0	0.0030036	0
	S.D	1.0831	0	4.26E-40	0	0.001052	0
	Best	672	0	26	0	0	0
	Worst	881	0	70	0	0	0
22	Median	799.5	0	44.5	0	0	0
	Mean	783.9333	0	47.5333	0	0	0
	S.D	57.3597	0	12.1335	0	0	0
	Best	14684	0	218	0	0	0
	Worst	43333	0	1215	0	0	0
23	Median	33456.5	0	614.5	0	0	0
	Mean	34109.1667	0	612.6	0	0	0
	S.D	6721.4537	0	257.6794	0	0	0
	Best	0	0	0	0	0	0
	Worst	1	0	0	0	0	0
24	Median	0	0	0	0	0	0
	Mean	0.066667	0	0	0	0	0
	S.D	0.25371	0	0	0	0	0
	Best	3195.407	2.79E-16	11.2992	5.14E-30	0.0077476	0
	Worst	5905.7553	2.67E-16	111.8415	8.86E-28	1.7702	0
25	Median	4865.1481	2.69E-16	47.7022	4.52E-29	0.072834	0
	Mean	4811.7969	2.72E-16	53.6457	1.21E-28	0.21006	0
	S.D	588.3249	8.51E-12	27.5951	1.76E-28	0.34752	0
	Best	1.2293	0.020580523	0	0	2.86E-05	0
	Worst	4.5856	0.154424436	0	0	0.00058994	0
26	Median	3.2952	0.063475712	0	0	0.00024973	0
	Mean	3.1853	0.069228159	0	0	0.00027632	0
	S.D	0.92982	0.036675834	0	0	0.00014538	0

Table 5.2 Results of the metaheuristics over benchmark test functions

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
27	Best	-106.7645	-106.7645	-106.7645	-106.7645	-106.7645	-106.7645
	Worst	-106.2736	-106.7645	-106.7645	-87.3109	-106.7645	-106.7393
	Median	-106.749	-106.7645	-106.7645	-106.7645	-106.7645	-106.7632
	Mean	-106.687	-106.7645	-106.7645	-105.4675	-106.7645	-106.7597
	S.D	0.13455	2.05E-16	1.09E-09	5.9359	6.27E-08	0.0070169
28	Best	0.0058429	0	0	0	1.08E-07	0
	Worst	0.22859	0	0	0	4.41E-06	0
	Median	0.058131	0	0	0	1.41E-06	0
	Mean	0.075826	0	0	0	1.60E-06	0
	S.D	0.059017	0	0	0	1.16E-06	0
29	Best	0.0019593	0	0	0	4.41E-08	0
	Worst	0.29095	0	0	0	3.66E-06	0
	Median	0.019745	0	0	0	5.74E-07	0
	Mean	0.037888	0	0	0	8.64E-07	0
	S.D	0.056996	0	0	0	9.09E-07	0
30	Best	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	Worst	-1.0297	-1.0316	-1.0316	-1.0316	-1.0316	-1.0088
	Median	-1.0315	-1.0316	-1.0316	-1.0316	-1.0316	-1.0296
	Mean	-1.0312	-1.0316	-1.0316	-1.0316	-1.0316	-1.0266
	S.D	5.65E-04	4.52E-16	4.65E-13	5.37E-09	1.77E-09	0.006205
31	Best	-2.0626	-2.0626	-2.0626	-2.0626	-2.0626	-2.0626
	Worst	-2.0623	-2.0626	-2.0626	-2.0626	-2.0626	-2.0626
	Median	-2.0626	-2.0626	-2.0626	-2.0626	-2.0626	-2.0626
	Mean	-2.0626	-2.0626	-2.0626	-2.0626	-2.0626	-2.0626
	S.D	0.000057938	9.03E-16	1.35E-10	2.70E-09	5.03E-07	1.02E-03
32	Best	-0.99989	-1	-1	-1	-1	-1
	Worst	-0.93623	-1	-1	-0.9362453	-1	-1
	Median	-0.99233	-1	-1	-1	-1	-1
	Mean	-0.98567	-1	-1	-0.9978748	-1	-1
	S.D	0.01849	0.00E+00	8.63E-13	0.011639957	7.17E-09	0
33	Best	-959.6403	-959.6407	-959.6407	-959.64066	-959.6407	-959.6407
	Worst	-887.0879	-959.6097	-894.5789	-786.522428	-718.1675	-959.6407
	Median	-959.1016	-959.6406	-959.6407	-959.64064	-959.6407	-959.6407
	Mean	-954.6817	-959.638708	-957.4719	-920.641634	-912.4829	-959.6407
	S.D	13.5893	0.006340094	11.8786	54.64965465	71.4234	2.04E-12
34	Best	0.36776	4.44E-16	0	0	1.55E-08	0
	Worst	4.2538	0.0173688	0	0	1.85E-06	0
	Median	1.3513	8.16E-15	0	0	4.32E-07	0
	Mean	1.6518	5.79E-04	0	0	5.55E-07	0
	S.D	1.0598	0.003171094	0	0	4.58E-07	0
35	Best	3	3	3	3	3	3.0001
	Worst	3.0031	3	3	3	3	16.6273
	Median	3.0002	3	3	3	3	3.0105
	Mean	3.0004	3	3	3	3	3.0185
	S.D	6.05E-04	1.79E-15	1.02E-15	1.31E-05	9.72E-09	0.023689
36	Best	-3.8627	-3.2618	-3.8628	-3.86277956	-3.8628	-3.8609
	Worst	-3.859	-3.0021	-3.8628	-3.85490064	-3.8628	-0.9528
	Median	-3.8622	-3.0428	-3.8628	-3.8628	-3.8628	-3.7741
	Mean	-3.8619	-3.8628	-3.8628	-3.8624	-3.8628	-3.7241
	S.D	0.00095848	4.723E-10	3.16E-15	0.002870273	4.72E-10	0.72518
37	Best	-3.0349	-3.0425	-3.0425	-3.04245754	-3.0425	-3.0425
	Worst	-2.9144	-2.98	-2.981	-2.91775065	-2.98	-2.83984
	Median	-2.9855	-3.0425	-3.0425	-3.04245429	-3.0425	-3.03345
	Mean	-2.9851	-3.028	-3.0404	-3.01605053	-3.028	-3.01605
	S.D	0.035303	0.026583	0.011219	0.037190414	0.026583	0.045706
38	Best	-19.2085	-19.2085	-19.2085	-19.2085	-19.2085	-19.2085
	Worst	-19.1928	-19.2085	-19.2085	-19.2042	-19.2085	-19.2085
	Median	-19.2072	-19.2085	-19.2085	-19.2085	-19.2085	-19.2085
	Mean	-19.2056	-19.2085	-19.2085	-19.2084	-19.2085	-19.2085
	S.D	0.0039289	1.2783E-08	4.32E-07	0.00077948	1.28E-08	0.0015E-09
39	Best	-0.67367	-0.67366	-0.67367	-0.67367	-0.67367	-0.67367
	Worst	-0.67365	-0.67367	-0.67367	-0.67367	-0.67367	-0.67189
	Median	-0.67367	-0.67367	-0.67367	-0.67367	-0.67367	-0.67367
	Mean	-0.67367	-0.67367	-0.67367	-0.67367	-0.67367	-0.67354
	S.D	4.0838E-06	8.25E-17	3.35E-16	1.24E-08	1.11E-11	0.00039502

Table 5.2 Results of the metaheuristics over benchmark test functions

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
	Best	8.2958E-06	3.18E-21	1.35E-31	1.43E-09	1.27E-10	5.47E-08
	Worst	0.0047618	1.08E-18	1.35E-31	3.07E-07	5.47E-08	0.12216
40	Median	0.00073654	2.56E-19	1.35E-31	6.21E-08	8.04E-09	0.0024452
	Mean	0.0012918	3.19E-19	1.35E-31	8.36E-08	1.29E-08	0.011027
	S.D	0.0013928	2.77E-19	6.68E-47	8.32E-08	1.31E-08	0.023757
	Best	8.8242	3.82E-16	1.3124	0.0065555	1.20E-05	0.0089157
	Worst	14.8547	1.18E-15	30.5618	0.063406	5.41E-05	0.075277
41	Median	11.8014	7.86E-16	5.7516	0.020171	2.20E-05	0.024699
	Mean	11.8403	7.86E-16	7.6587	0.024756	2.55E-05	0.025745
	S.D	4.0177	1.61E-16	6.5852	0.013532	1.10E-05	0.013262
	Best	2.1049	5.0164E-16	29.6125	1.41E-05	0.0001365	0.15734
	Worst	5.7544	1.4295E-15	519068.5251	0.80499	0.00053382	2.9718
42	Median	3.3085	8.7177E-16	522.7366	0.29864	0.0002889	1.3694
	Mean	3.5478	8.60768E-16	30170.4933	0.29976	0.00028766	1.6593
	S.D	7.0145	1.79894E-16	98971.8372	0.19359	9.07E-05	1.2139
	Best	7.67E-02	1.31E-08	7.91E-06	9.38E-06	4.63E-07	0.017449
	Worst	3.2113	0.011847851	0.00096267	1.5743	0.01093	2.9394
43	Median	1.1133	0.000954721	0.00019502	0.00027899	0.0005889	0.074672
	Mean	1.2307	0.003526435	0.00023581	0.05431	0.0038414	0.94867
	S.D	0.84764	0.001604834	0.00023964	0.2871	0.0050794	0.85019
	Best	2.73E-02	5.82E-05	0.00019968	0.00012909	1.98E-08	0.00031389
	Worst	2.3947	0.005481231	0.010521	0.88183	0.0016547	8.6957
44	Median	0.30682	0.000318945	0.0015656	0.0011625	0.0002464	0.81461
	Mean	5.13E-01	0.00013892	0.0021292	0.10797	0.00029232	0.067372
	S.D	5.10E-01	0.00013892	0.0021506	0.25769	0.000372	1.8389
	Best	0.00010208	1.45E-02	9.19E-13	9.50E-09	2.81E-08	8.30E-16
	Worst	0.33693	5.2447	1.32E-07	6.30E-07	1.20E-05	4.17E-06
45	Median	0.040306	4.5208	2.60E-10	2.08E-07	1.58E-06	1.70E-09
	Mean	0.065738	1.8509	2.60E-09	2.41E-07	2.34E-06	2.61E-07
	S.D	0.077089	14.1462	2.41E-08	1.88E-07	2.91E-06	8.00E-07
	Best	0.90001	0.9	0.9	0.9	0.9	0.9
	Worst	1.0004	0.9	0.90001	1.0001	0.91505	0.9
46	Median	0.90536	0.9	0.9	0.9	0.9	0.9
	Mean	0.91429	0.9	0.9	0.90667	0.90116	0.9
	S.D	0.023247	4.52E-16	1.66E-06	0.025383	0.0030237	4.52E-16
	Best	0.00052928	9.3699E-06	9.29E-21	1.64E-10	1.37E-10	0.00078669
	Worst	0.057101	0.083267	8.87E-08	0.0074155	0.043267	0.83767
47	Median	0.025498	0.000019209	1.32E-11	0.0037077	1.92E-08	0.044902
	Mean	0.026035	0.0014423	4.57E-09	0.0037077	0.0014423	0.13661
	S.D	0.019169	0.0078994	1.65E-08	0.0037711	0.0078994	0.20926
	Best	0.0008569	0.000003268	4.42E-17	2.95E-20	1.33E-10	3.52E-249
	Worst	19.525	0.0009642	1.24E-12	1.30E-11	2.96E-07	2.34E-10
48	Median	0.57055	0.0029993	1.64E-15	3.88E-13	3.00E-08	5.25E-73
	Mean	1.8298	0.000589524	4.81E-14	1.46E-12	5.91E-08	7.81E-42
	S.D	3.8376	9.80078E-06	2.26E-13	2.79E-12	8.01E-08	4.28E-14
	Best	-186.7306	-186.7309	-186.7309	-186.7309	-186.7309	-186.7309
	Worst	-176.1507	-186.7309	-186.7301	-186.5771	-186.7309	-121.7144
49	Median	-185.4696	-186.7309	-186.7308	-186.7308	-186.7309	-186.544
	Mean	-184.4844	-186.7309	-186.7307	-186.7215	-186.7309	-180.9683
	S.D	2.7483	8.88E-06	0.00018617	0.031254	1.55E-06	0.0015698
	Best	-29.6745	-29.6759	-29.6759	-29.6759	-29.6759	-29.6759
	Worst	-28.8867	-29.6759	-29.6758	-29.6758	-29.2371	-29.6758
50	Median	-29.6007	-29.6759	-29.6759	-29.6759	-29.6759	-29.6788
	Mean	-29.5224	-29.6759	-29.6759	-29.6759	-29.6613	-29.6786
	S.D	0.19679	3.61E-15	2.17E-05	1.96E-05	0.080107	2.4181E-17
	Best	-25.6987	-25.7418	-25.7418	-25.7418	-25.7418	-25.7418
	Worst	-22.4133	-25.7416	-25.7416	-21.3887	-25.7418	-25.7407
51	Median	-25.4185	-25.7418	-25.7418	-25.7418	-25.7418	-25.7418
	Mean	-25.0985	-25.74177	-25.7418	-25.5946	-25.7418	-25.7418
	S.D	0.83683	1.45E-14	3.09E-05	0.79445	1.80E-07	0.000326
	Best	3.1901	5.04E-08	3.0672	4.60E-41	0.0049316	0
	Worst	5.6217	1.83E-05	9.2612	0.00088014	0.021716	0
52	Median	4.1841	8.59E-07	6.2124	4.04E-38	0.0092847	0
	Mean	4.9583	1.76E-06	6.2209	4.66E-05	0.010317	0
	S.D	1.4454	3.35E-06	1.7611	0.00017085	0.0039505	0

Table 5.2 Results of the metaheuristics over benchmark test functions

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
	Best	0.0014994	0	0	0	6.90E-08	0
	Worst	0.35754	0	0	0	7.60E-06	0
53	Median	0.050016	0	0	0	1.15E-06	0
	Mean	0.10545	0	0	0	2.06E-06	0
	S.D	0.11116	0	0	0	1.89E-06	0
	Best	7.1698E-07	4.70E-20	0	1.63E-10	2.81E-12	3.98E-10
	Worst	0.00089499	3.53E-17	0	2.66E-07	5.20E-09	0.039862
54	Median	0.000077554	5.50E-18	0	4.79E-08	1.65E-09	0.000117
	Mean	0.00017393	8.57E-18	0	6.43E-08	2.20E-09	0.0016352
	S.D	0.00023355	9.23E-18	0	6.59E-08	1.56E-09	0.0072483
	Best	0.39789	0.39788	0.39789	0.39789	0.39789	0.3989
	Worst	0.39935	0.39788	0.39789	0.39797	0.39789	0.3989
55	Median	0.39802	0.39788	0.39789	0.39789	0.39789	0.3989
	Mean	0.39816	0.39788	0.39789	0.39789	0.39789	0.3989
	S.D	0.00038302	0	1.59E-11	1.58E-05	1.17E-09	0
	Best	3.3749E-07	9.90E-21	1.12E-49	0	1.02E-10	0
	Worst	0.0019971	1.91E-18	1.33E-34	0	1.44E-08	0
56	Median	0.00013971	3.33E-19	1.12E-45	0	5.33E-09	0
	Mean	0.00038729	5.01E-19	4.44E-36	0	5.28E-09	0
	S.D	0.00054741	4.62E-19	2.42E-35	0	3.95E-09	0
	Best	-1.8013	-1.80130341	-1.8013	-1.8013	-1.8013	-1.8013
	Worst	-1.801	-1.80130341	-1.8013	-1.8013	-1.8013	-0.99999
57	Median	-1.8013	-1.80130341	-1.8013	-1.8013	-1.8013	-1.7879
	Mean	-1.8013	-1.80130341	-1.8013	-1.8013	-1.8013	-1.6392
	S.D	6.79E-05	9.03E-16	9.03E-16	8.82E-07	6.23E-10	0.29461
	Best	-3.9775	-4.68765818	-4.6861	-4.6876	-4.6877	-3.9969
	Worst	-3.0457	-4.68765818	-4.4941	-3.5992	-4.1684	-1.701
58	Median	-3.4219	-4.68765818	-4.6438	-4.4959	-4.5377	-3.3491
	Mean	-3.4417	-4.68765818	-4.6447	-4.4262	-4.5629	-3.2611
	S.D	2.60E-01	2.61E-15	0.039355	0.32476	0.11493	0.49051
	Best	-5.4708	-9.66015172	-8.5604	-9.3067	-9.5515	-9.2284
	Worst	-3.8777	-9.66015172	-6.9282	-5.5648	-7.138	-3.7357
59	Median	-4.5549	-9.66015172	-7.5294	-8.0418	-9.0418	-5.8739
	Mean	-4.5434	-9.66015172	-7.5887	-7.9528	-8.8045	-5.6797
	S.D	0.41908	0	0.41986	0.93142	0.6493	0.86726
	Best	0.024484	5.68E-14	0	0	4.94E-10	0
	Worst	2.151	1.40E-11	0	0	2.06E-07	0
60	Median	1.076	1.71E-13	0	0	5.18E-08	0
	Mean	1.1077	8.83E-13	0	0	5.99E-08	0
	S.D	0.55972	2.76E-12	0	0	5.08E-08	0

Table 5.3 summarizes the results and the comparison with other metaheuristics. The table also depicts the number of test functions that had been solved via NPO. The symbol '+' represents the number of test functions where NPO exhibited better results, while '-' denotes the worst results, '=' reflects both algorithms with similar good or bad results, '*' refers to the number of test functions where NPO reached the optimal solution. From the Table, it is evident that NPO has successfully outperformed the other metaheuristics by 86.6%.

Table 5.3 Summarized comparison results of NPO verses other algorithms

Tests	vs. PSO			vs. ABC			vs. GWO			vs. FFA			vs. FPA			Opt.	
	+	-	=	+	-	=	+	-	=	+	-	=	+	-	=	*	-
U-N U-S	21	0	4	5	0	18	2	0	23	12	2	11	11	2	12	23	2
M-N M-S	14	2	19	7	2	26	7	0	28	9	1	25	2	1	32	29	6
All Tests	35	2	23	12	2	44	9	0	51	21	3	36	13	3	44	52	8

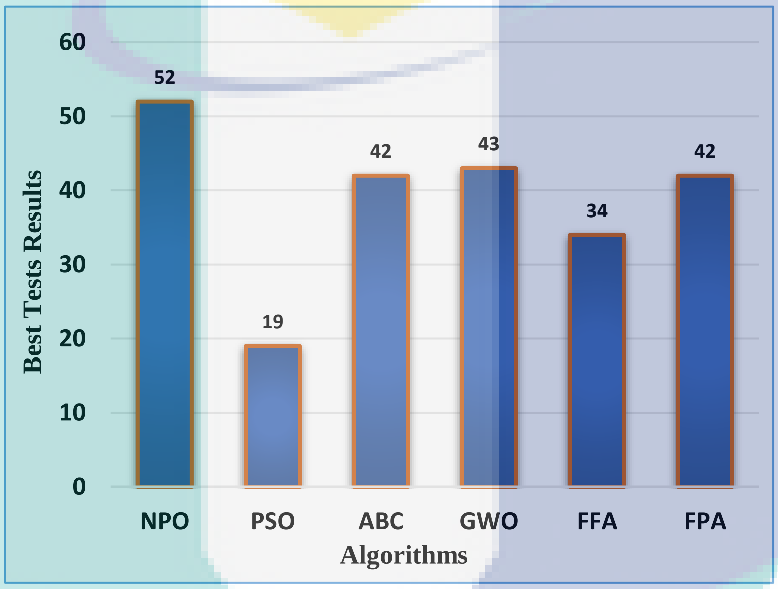


Figure 5.2 The results of all tests

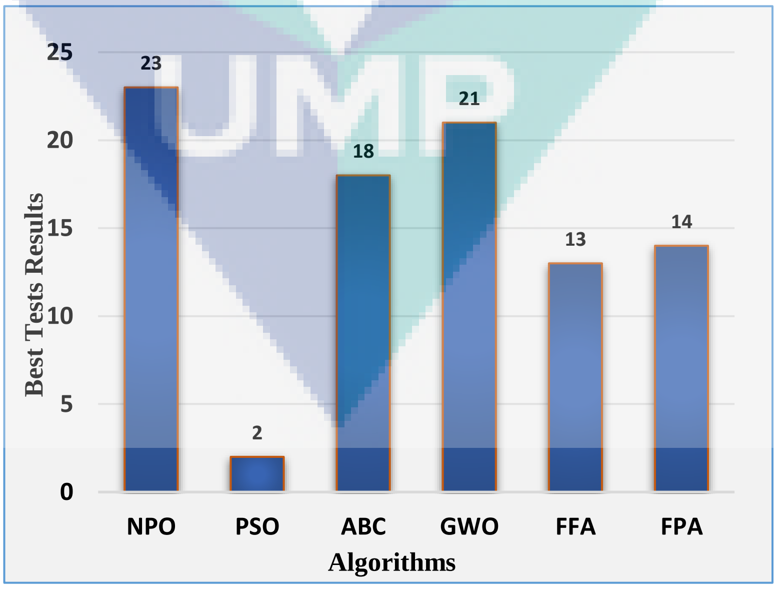


Figure 5.3 The results of unimodal tests

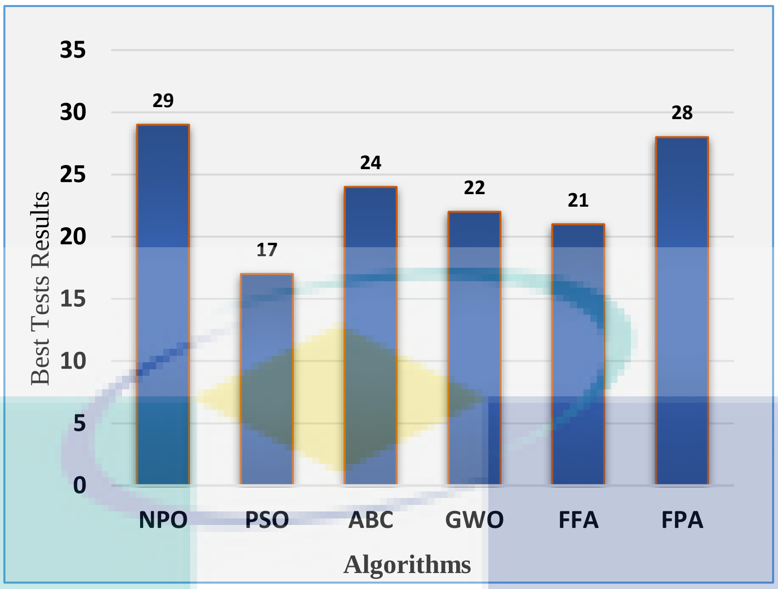


Figure 5.4 The results of multimodal tests

All algorithms have been ranked statistically based on Mean Absolute Error (MAE). MAE determines which algorithm has the lowest difference (i.e. error) to the optimal solution. It can be calculated by using the following equation:

$$MAE = \frac{\sum_{i=1}^{N} | m_i - k_i |}{N} \qquad 4.1$$

where m_i indicates the mean of optimal values, k_i is the corresponding global optimal value, and N represents the number of samples. In our case, N denotes the number of test functions. Table 5.4 shows the ranking of the algorithms based on MAE equation. It is clear that NPO outperformed other algorithms, it has a very small difference between the mean values of all test with the optimal values. FPA and PSO have very large MAEs, because they failed to solve several test functions. These test functions obtained very large mean values with FPA in the last position.

Table 5.4 The rank of algorithm based on MAE

Unimodal			Multimodal			All Tests		
Rank	**Alg.**	**MAE**	**Rank**	**Alg.**	**MAE**	**Rank**	**Alg.**	**MAE**
1	**NPO**	**0.013871**	1	**ABC**	**0.063359**	1	**NPO**	**0.29188916**
2	GWO	0.865159	2	NPO	0.490473	2	GWO	1.08097118
3	FFA	0.886671	3	GWO	1.178871	3	FFA	1.177124
4	ABC	11.72098	4	FFA	1.468787	4	ABC	4.9207
5	FPA	8.86E+11	5	FPA	152.4946	5	FPA	3.69083E+11
6	PSO	1.84E+15	6	PSO	168.9485	6	PSO	7.66436E+16

Although the statistical results presented in Table 5.3 provide a first insight into the performance of NPO, a pair-wise statistical test is typically used for a better comparison. For this purpose, by using the results obtained from 30 runs of each algorithm, a Wilcoxon Signed-Rank Test is performed with a statistical significance value $\propto = 0.05$. The null hypothesis H_0 for this test is: "There is no difference between the median of the solutions produced by algorithm A and the median of the solutions produced by algorithm B for the same benchmark problem". i.e., median (A) = median (B). To determine whether algorithm A reached a statistically better solution than algorithm B, or if not, whether the alternative hypothesis is valid, the size of the ranks provided by the Wilcoxon Signed-Rank Test (i.e., T+, and T-)) are examined.

In **Table 5.5**, the statistical pair-wise results of the NPO algorithm compared to those of other algorithms are given. In this table, + indicates the positive ranks, while – indicates the negative ranks, finally, = indicates the equal ranks. In the z-distribution column, the asterix (*) indicates that a p-value of less than 0.05, which means there is a significant difference between the two algorithms in that test. The legends used in this test are:

b- The sum of negative ranks equals the sum of positive ranks.

c- Based on positive ranks.

d- Based on negative ranks

Table 5.5 The Wilcoxon Signed Rank Test

f_n	NPO vs. PSO				NPO vs. ABC				NPO vs. FPA				NPO vs. GWO				NPO vs. FA			
	+	-	=	*z*	+	-	=	*z*	+	-	=	*z*	+	-	=	*z*	+	-	=	*z*
f_1	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b
f_2	24	6	0	-2.705, d*	0	30	0	-4.782, c*	0	30	0	-4.782, c*	3	27	0	-2.993, c*	0	30	0	-4.782, c*
f_3	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_4	30	0	0	-4.782, d*	25	5	0	-2.023, d*	0	0	30	0.000, b	0	0	30	0.000, b	8	0	22	-2.828, d*
f_5	1	29	0	-4.762, d*	1	29	0	-4.762, c*	0	30	0	-4.782, c*	1	29	0	-4.762, c*	0	30	0	-4.782, c*
f_6	30	0	0	-4.782, d*	30	0	0	-4.286, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_7	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_8	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b	30	0	0	-4.782, d*
f_9	30	0	0	-4.782, d*	24	0	6	-4.286, d*	0	0	30	0.000, b	0	0	30	0.000, b	30	0	0	-4.782, d*
f_{10}	30	0	0	-4.782, d*	20	0	10	-3.922, d*	3	0	27	-1.633, d	7	0	23	-2.460, d*	0	30	0	-4.867, c*
f_{11}	0	30	0	-4.782, c*	2	0	28	-1.414, d	0	0	30	0.000, b	0	0	30	0.000, b	4	26	0	-2.651, c*
f_{12}	30	0	0	-4.782, d*	2	0	28	-1.414, d	0	0	30	0.000, b	19	0	11	-4.264, d*	0	0	30	0.000, b
f_{13}	30	0	0	-4.782, d*	7	0	23	-2.460, d*	0	2	28	-1.414, c	30	0	0	-4.782, d*	12	1	17	-3.197, d*
f_{14}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{15}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	0	0	30	0.000, b	30	0	0	-4.783, d*
f_{16}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{17}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{18}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{19}	30	0	0	-4.782, d*	30	0	0	-4.783, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{20}	30	0	0	-4.782, d*	30	0	0	-4.783, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{21}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	0	0	30	0.000, b	30	0	0	-4.782, d*
f_{22}	30	0	0	-4.782, d*	0	0	30	0.000, b	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b
f_{23}	30	0	0	-4.782, d*	30	0	0	0.000, b	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b
f_{24}	2	0	28	-1.414, d	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	-4.782, b
f_{25}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{26}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b	30	0	0	-4.782, d*
f_{27}	5	0	25	-1.624, d	0	20	10	-4.782, c*	0	20	10	-4.782, c*	0	20	10	-4.782, c*	0	20	10	-4.782, c*
f_{28}	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b	30	0	0	-4.782, d*
f_{29}	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b	0	30	0	-4.782, c*
f_{30}	0	3	27	-1.633, c	0	3	27	-1.633, c	0	3	27	-1.633, c	0	3	27	-1.633, c	0	3	27	-1.633, c
f_{31}	19	11	0	-1.820, d	0	4	26	-1.841, c*	0	30	0	-5.201, c*	0	4	26	-1.841, c	0	30	0	-5.201, c*
f_{32}	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b	1	0	29	-1.000, d	0	0	30	0.000, b
f_{33}	30	0	0	-4.782, d*	17	0	13	-3.640, d*	0	0	30	0.000, b	16	0	14	-3.520, d*	13	17	0	-1.677, c
f_{34}	30	0	0	-4.782, d*	30	0	0	-4.783, d*	0	0	30	0.000, b	0	0	30	0.000, b	30	0	0	-4.782, d*

Table 5.5 The Wilcoxon Signed Rank Test

f_n	NPO vs. PSO				NPO vs. ABC				NPO vs. FPA				NPO vs. GWO				NPO vs. FA			
	+	-	=	z	+	-	=	z	+	-	=	z	+	-	=	z	+	-	=	z
f_{35}	1	29	0	-4.741, c*	0	30	0	-4.782, c*	0	30	0	-4.782, d*	1	29	0	-4.165, c*	0	30	0	-4.782, c*
f_{36}	1	29	0	-4.762, c*	1	29	0	-4.762, c*	1	29	0	-4.762, c*	0	30	0	-4.782, c*	0	30	0	-4.782, c*
f_{37}	26	4	0	-3.445, d*	16	14	1	-0.086, d	5	24	1	-3.060, c*	14	15	1	-0.076, d	10	19	0	-1.373, c
f_{38}	26	4	0	-3.795, d*	14	16	1	-1.970, c*	0	13	17	-3.186, c*	12	17	1	-1.990, c*	0	13	17	-3.186, c*
f_{39}	16	14	0	-0.175, d	0	30	0	-4.852, c*	0	30	0	-4.852, c*	8	8	14	0.000, b	0	22	8	-4.236, c*
f_{40}	12	18	0	-2.129, c*	0	30	0	-4.782, c*	0	30	0	-4.782, c*	0	30	0	-4.782, c*	0	29	1	-4.782, c*
f_{41}	30	0	0	-4.782, d*	0	30	0	-4.782, c*	30	0	0	-4.782, d*	16	14	0	-1.635, d	0	30	0	-4.782, c*
f_{42}	30	0	0	-1.903, d	0	30	0	-4.782, c*	30	0	0	-4.782, d*	6	24	0	-3.898, c*	0	30	0	-4.782, c*
f_{43}	18	12	0	-1.306, d	4	26	0	-3.445, c*	0	30	0	-4.782, c*	2	28	0	-3.980, c*	0	30	0	-4.782, c*
f_{44}	21	9	0	-1.244, d	4	26	0	-2.651, c*	1	29	0	-4.762, c*	2	28	0	-3.939, c*	0	30	0	-4.782, c*
f_{45}	30	0	0	-4.782, d*	29	1	0	-4.474, d*	10	20	0	-1.306, c	29	1	0	-4.474, d*	30	0	0	-4.782, d*
f_{46}	30	0	0	-4.782, d*	0	0	30	0.000, b	1	0	29	-1.000, d	0	0	30	0.000, b	9	0	21	-2.666, d*
f_{47}	21	9	0	-2.849, c*	4	26	0	-2.651, c*	0	30	0	-4.782, c*	3	27	0	-4.391, c*	1	29	0	-4.515, c*
f_{48}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	29	1	0	-4.165, d*	29	1	0	-4.165, d*	30	0	0	-4.782, d*
f_{49}	30	0	0	-1.903, d	3	27	0	-4.268, c*	1	29	0	-4.762. c*	3	27	0	-4.268, c*	0	29	1	-4.703, c*
f_{50}	30	0	0	-4.782, d*	0	17	13	-3.727, c*	1	16	13	-3.219, c*	3	15	12	-1.686, c	1	16	13	-2.893, c*
f_{51}	29	1	0	-4.165, d	3	4	23	-0.689, c	9	4	17	-0.825, d*	16	3	11	-2.241, d*	0	4	26	-1.841, c*
f_{52}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*
f_{53}	30	0	0	-4.782, d*	0	0	30	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b	30	0	0	-4.782, d*
f_{54}	1	29	0	-4.741, c*	0	30	0	-4.782, c*	0	30	0	-4.782, c*	0	30	0	-4.782, c*	0	30	0	-4.782, c*
f_{55}	3	27	0	-4.659, c*	0	30	0	-5.477, c*	0	30	0	-5.477, c*	0	30	0	-5.477, c*	0	0	30	0.000, b
f_{56}	30	0	0	-4.782, d*	30	0	0	-4.782, d*	30	0	0	-4.782, d*	0	0	30	0.000, b	30	0	0	-4.782, d*
f_{57}	2	28	0	-4.659, c*	0	30	0	-4.784, c*	0	30	0	-4.784, c*	0	30	0	-4.784, c*	0	30	0	-4.784, c*
f_{58}	7	23	0	-1.841, c	0	30	0	-4.782, c*	0	30	0	-4.782, c*	1	29	0	-4.618, c*	0	30	0	-4.782, c*
f_{59}	24	6	0	-4.083, d*	0	30	0	-4.782, c*	1	29	0	-4.350, c*	1	29	0	-4.762, c*	1	29	0	-4.741, c*
f_{60}	30	0	0	-4.782, d*	29	0	1	0.000, b	0	0	30	0.000, b	0	0	30	0.000, b	30	0	0	-4.782, d*

The previous table displayed the Wilcoxon Signed Rank test for comparison of the 30 runs of each metaheuristic. The table can be summarized as follows:

- NPO vs. PSO: The test indicates that there are more significant negative ranks (N = 50) rather than significant positive ranks (P = 10). This means that the median of NPO is more than median of PSO, in other words, the H_0 is rejected and the NPO has superior performance and has outperformed PSO.

- NPO vs. ABC: The test indicates that there are more significant negative ranks (N = 26) rather than significant positive ranks (P = 23). This means that the median of NPO is more than median of PSO, in other words, the H_0 is rejected and NPO is statistically better than ABC algorithm.

- NPO vs. FPA: The test indicates that there are more significant negative ranks (N = 23) rather than significant positive ranks (P = 22). This means that the median of NPO is more than median of PSO, in other words, the H_0 is rejected and NPO has better performance than FPA.

- NPO vs. GWO: The test indicates that there more significant negative ranks (N = 21) rather than significant positive ranks (P = 20). This means that the median of NPO is more than median of PSO, in other words, the H_0 is rejected and the NPO is better than GWO, however, there are 19 tests where both have equal results.

- NPO vs. FFA: The test indicates that there more significant negative ranks (N = 28) rather than significant positive ranks (P = 26). This means that the median of NPO is more than median of PSO, in other words, the H_0 is rejected, and the NPO statistically is better than FFA.

5.3.2 NPO for Large-Scale Problems

To test the robustness of NPO over large-scale problems with different dimensions sizes, several continuous benchmark functions were chosen and used with medium (100), large (500), and very large (1000 and 2000) number of decision variables (dimensions). The results of the 30 runtimes (best, worst, median, mean and standard deviation) are compared to those of the mentioned metaheuristics. The test functions used in this section are 18 problems. These problems are ($f3, f14, f15, f16, f17, f18, f19, f20, f21, f22, f23, f25, f26, f34, f41, f42, f52$, and $f60$) . The main difference between these test functions and the rest is that these functions have dynamic number of variables, thus, these functions can be used with the mentioned above dimensions. The results are presented in Table 5.6, Table 5.7, Table 5.8, and Table 5.9 respectively.

The results have clearly showed that NPO has the ability to handle the large-scale problems. NPO has attained the best results in 16 out of 18 test functions, while the other algorithms have failed with most of these tests, especially with the very large problems (i.e., number of dimensions = 2000). The multi-swarm structure of NPO provides stable performance in terms of the scalability, and outperforms the other algorithms from the literature. The exploration part of NPO helps the algorithm to explore a wide area in the search space, and avoid trapping in the local optima.

Table 5.6 Results for Large-scale problems, D = 100

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
3	Best	436.081	4.014E-04	0	1.5659E-67	0.00143	0
	Mean	745.205	2.474E-03	4.4369E-10	9.1041E-65	0.0022	0
	S.D	359.351	3.047E-03	1.1389E-09	1.2454E-65	8.316E-04	0
14	Best	1334.055	1254.458	541.7490	7.1317E-08	0.9161	6.710E-277
	Mean	1429.126	1365.845	987.7509	1.9303E-05	23.0758	1.53E-39
	S.D	54.767	84.4225	243.5670	3.4657E-05	18.5865	7.3353E-39
15	Best	0.10286	4.278E-10	5.4514e-12	0	2.9247E-21	0
	Mean	0.21673	1.215E-06	5.8140e-07	0	6.3895E-20	0
	S.D	0.11339	4.514E-06	1.5695e-06	0	2.3877E-19	0
16	Best	664.939	2.0912	2.1086	1.6465E-04	0.0113	7.32E-07
	Mean	773.376	2.5277	5.1964	9.3399E-04	0.0239	1.39E-05
	S.D	108.421	0.3258	2.7691	4.3166E-04	0.0062	1.28E-05
17	Best	3979.665	1.7350	500.2493	2.3363E-37	1.9038	0
	Mean	3270.285	3.7836	590.1090	9.1332E-37	2.1659	0
	S.D	1770.391	3.6024	73.2246	5.4085E-37	0.2972	0
18	Best	81.7736	86.4876	23.9367	6.0793E-12	0.8033	0
	Mean	85.0544	89.1781	27.6855	1.1477E-08	3.9197	0
	S.D	2.7092	1.6063	2.4035	3.6298E-08	4.8113	0
19	Best	4.335E+129	0.0049	1.4386E+49	1.3413E-37	66.0143	0
	Mean	6.860E+129	0.0110	3.1938E+65	9.9048E-37	148.4295	0
	S.D	9.423E+130	0.0097	8.5272E+65	5.4157E-37	116.4210	0
20	Best	3.0868E+09	7.715E-16	5296.51	8.2247E-201	4.928E-13	0
	Mean	6.8301E+09	1.220E-14	43319.74	8.4557E-186	1.092E-12	0
	S.D	3.2105E+09	1.830E-14	33137.14	2.4784E-187	1.246E-12	0
21	Best	152314.109	0.0064	5835.941	1.6252E-64	0.040309	0
	Mean	159666.025	0.0117	7814.524	6.4274E63	0.044081	0
	S.D	7207.4705	0.0048	1593.364	1.1686E-62	0.0032	0
22	Best	3233	0	513	0	0	0
	Mean	3270.6	0	617.33	0	0	0
	S.D	48.726	0	56.049	0	0	0
23	Best	158470	2	6641	0	0	0
	Mean	168557.2	6.4511	9546.314	0	0	0
	S.D	7004.646	1.8542	2047.47	0	0	0
25	Best	72373.335	5.128E-09	2793.84	3.0859E-65	1.3207	0
	Mean	7825.9311	6.148E-08	4353.14	3.0494E-63	3.2474	0
	S.D	5514.437	1.147E-08	836.445	4.8647E-63	1.2411	0
26	Best	0.120141	0.4537	0	0	3.4214E-09	0
	Mean	0.591568	1.5798	0	0	1.3468E-04	0
	S.D	0.442324	1.0513	0	0	7.5581E-05	0
34	Best	0.14781	4.151E-06	0	0	3.14E-08	0
	Mean	0.23108	0.0024	0	0	1.45E-07	0
	S.D	0.08286	0.0050	0	0	1.86E-07	0
41	Best	5.533E+08	0.47845	15.3689	0.13323	7.434E-05	0.8788
	Mean	7.902E+08	0.44885	32366.04	0.23647	9.956E-05	0.9922
	S.D	1.436E+08	0.00474	59623.47	0.06456	2.184E-05	0.0578
42	Best	1.718E+09	1.8745	2.499E+05	5.0365	0.00285	8.3533
	Mean	1.864E+09	2.8471	2.057E+06	6.0282	0.00344	9.8193
	S.D	1.347E+08	0.0576	2.215E+06	0.4857	5.634E-04	0.28
52	Best	185.5875	0.0822	30.0075	6.0308E-38	0.0552	0
	Mean	204.521	0.1556	39.6843	5.5211E-36	0.3184	0
	S.D	9.0134	0.0601	5.0509	3.0247E-35	0.2329	0
60	Best	0.03714	13.1517	0	0	4.9966E-10	0
	Mean	1.02581	16.6879	0	0	1.1010E-08	0
	S.D	0.57207	2.4709	0	0	1.1926E-08	0

Table 5.7 Results for Large-scale problems, D = 500

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
	Best	6.427E-04	0.1745	32435.141	2.3014E-33	115.5513	0
3	Mean	0.0264079	1.6847	47834.784	7.2738E-29	277.7533	0
	S.D	0.0431787	0.9845	14199.052	3.9080E-28	170.7327	0
	Best	3.55412	7726.247	13357.241	1337.803	4021.204	1.09E-93
14	Mean	4.77746	8094.705	4.323E+16	2035.742	4229.626	1.23E-21
	S.D	0.85447	246.1136	1.514E+16	3506.202	85.069	6.677E-21
	Best	0.08494	0.0077	1.746E-10	1.7013E-69	1.1488E-07	0
15	Mean	0.16394	0.0247	1.014E-05	1.4947E-11	1.5689E-06	0
	S.D	0.06386	0.0180	3.688E-05	7.9674E-11	1.8213E-06	0
	Best	21895.22	9.5551	3945.361	0.0015	1.3354	2.27E-07
16	Mean	23222.46	13.2630	4576.369	0.0034	1.6052	2.13E-05
	S.D	1034.118	4.3713	358.529	0.0012	0.2452	2.10E-05
	Best	177849.305	2281.812	3945.361	1.0952E-17	28.0671	0
17	Mean	181129.776	2429.345	4576.369	2.4946E-17	37.5265	0
	S.D	235.896	130.7143	358.529	1.0523E-17	14.2496	0
	Best	90.0878	98.0309	32.087	32.3180	78.0813	0
18	Mean	91.6079	98.2855	36.716	46.7162	79.1870	0
	S.D	1.07263	0.1744	3.2361	6.1147	1.3015	0
	Best	-	2.5478	2646.036	2.3985E-18	INF	0
19	Mean	-	2.2331	INF	2.2662E+102	INF	0
	S.D	-	0.4784	NaN	1.2387E+103	NaN	0
	Best	6.497E+10	3479.34	391761.112	6.5694E-89	1.4661	0
20	Mean	6.775E+10	2.346E+08	3146682.952	2.3123E-72	2.4460	0
	S.D	1.808E+10	4.603E+08	3270352.318	8.7302E-72	1.2895	0
	Best	15160.206	17730.447	495913.721	3.4875E-31	0.2541	0
21	Mean	19520.951	34161.687	709919.845	1.4785E-30	0.6185	0
	S.D	3517.773	10941.154	10100.475	2.6478E-30	0.0561	0
	Best	2.076E+07	8	4076	0	4	0
22	Mean	4.753E+07	22.842	4471.00	0	31.3334	0
	S.D	3.544E+07	24.484	298.707	0	38.2165	0
	Best	7.040E+30	18	46651	0	16	0
23	Mean	1.574E+30	24.1482	72006.133	0	17.00	0
	S.D	3.101E+30	3.4845	16276.273	0	1.00	0
	Best	2.267E+06	0.00148	126351.566	9.5724E-31	1880.0944	0
25	Mean	2.318E+06	0.02854	173852.407	2.7055E-30	2248.810	0
	S.D	5.125E+04	0.9453	22891.88	1.6336E-30	581.7029	0
	Best	0.03449	2.7839	0	0	1.594E-04	0
26	Mean	0.27365	3.4535	0	0	0.0011	0
	S.D	0.14307	0.5038	0	0	5.403E-04	0
	Best	0.05649	199.0954	0	0	1.8316e-08	0
34	Mean	0.17311	278.6552	0	0	4.8136e-06	0
	S.D	0.09765	61.7961	0	0	5.6278e-06	0
	Best	5.483E+09	4.6874	199170.663	0.70820	8.4167	1.0599
41	Mean	5.734E+09	8.9965	12613.77	0.75102	13.3083	1.1182
	S.D	1.877E+08	2.9984	14376.24	0.02623	3.6139	0.0232
	Best	9.850E+09	39.9985	1.097E+07	44.4242	160.9658	49.385
42	Mean	1.104E+10	42.8541	3.525E+07	45.7044	187.347	49.435
	S.D	7.341E+08	1.3147	2.387E+07	0.58574	17.9665	0.0165
	Best	1118.486	104.5457	281.22	1.3988E-18	20.0697	0
52	Mean	1136.983	112.5482	307.91	5.6327E-05	32.3101	0
	S.D	22.4322	5.4539	16.73	2.8410E-04	5.4495	0
	Best	0.006272	1434.661	0	0	5.2875e-08	0
60	Mean	0.024489	1487.747	0	0	0.1159	0
	S.D	0.021632	54.4633	0	0	0.2828	0

Table 5.8 Results for Large-scale problems, D = 1000

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
	Best	1.884E+14	8.2845	73321	2.9811E-23	92233	0
3	Mean	6.724E+14	13.8471	14721	3.4570E-07	11402	0
	S.D	5.593E+14	1.4852	62138	1.8733E-06	16272	0
	Best	8.780E+20	16471.845	4.254E+20	5.0565E+03	10289.103	1.4641E-29
14	Mean	1.247E+21	17093.965	7.153E+20	6.1382E+03	10659.488	2.458E-08
	S.D	4.426E+20	170.8093	4.710E+20	8.1277E+02	365.8192	5.145E-16
	Best	0.02814847	0.1618	1.0827E-12	4.104E-17	1.15E-07	0
15	Mean	0.14568460	0.5772	8.8026E-07	5852E-07	3.02E-06	0
	S.D	0.08088675	0.3346	2.5222E-06	1.5881E-06	4.63E-06	0
	Best	85741.0402	10.2358	579.315	0.0024	36.8031	7.56E-07
16	Mean	94055.5519	15.4248	1084.22	0.0054	45.7973	1.92E-05
	S.D	66473.3895	2.8910	3050.26	0.0017	8.3942	1.31E-05
	Best	36546.099	1522.404	8141.30	2.1143E-13	1019.825	0
17	Mean	36820.109	1679.982	96368.197	3.3337E-13	1064.665	0
	S.D	217.385	1215.932	689.673	7.6509E-14	229.1318	0
	Best	93.496	99.0736	35.6192	64.2613	90.8359	0
18	Mean	94.885	99.2117	39.6581	70.38787	91.7011	0
	S.D	0.8984	0.1038	2.5562	3.43506	0.4292	0
	Best	-	4.9584	1002.001	7.8935E+172	INF	0
19	Mean	-	5.1845	INF	3.8723E+223	INF	0
	S.D	-	0.2748	NaN	INF	NaN	0
	Best	1.441E+11	3.144E+10	1.9020E+06	1.2871E-62	2.5048E+06	0
20	Mean	1.552E+11	5.241E+10	9.8527E+06	1.2074E-40	6.4941E+06	0
	S.D	7.892E+09	1.356E+10	8.5741E+06	6.5674E-40	1.7085E+06	0
	Best	1.805E+06	6.0546E+05	1.089E+05	3.0443E-24	163632	0
21	Mean	1.870E+06	6.4537E+05	1.581E+05	8.4606E-24	181548	0
	S.D	4.334E+04	3.7818E+04	2.745E+05	4.3937E-24	7976.2	0
	Best	3520	34.8415	6880	0	9488	0
22	Mean	35816.40	39.8441	9510.4566	0	10089.15	0
	S.D	385.927	10.8475	8414.23	0	251.9855	0
	Best	1841577	52.8475	117894	0	160058	0
23	Mean	1876739.44	61.9481	15286.166	0	185726	0
	S.D	25551.224	16.9478	23709.350	0	12328	0
	Best	90849918.854	0.9385	5.251E+05	6.4287E-24	725113	0
25	Mean	93230340.852	1.0014	7.692E+05	3.1608E-23	836591	0
	S.D	21582519.451	0.0014	1.212E+05	1.4596E-23	41979	0
	Best	0.1248347549	0.7631	0	0	2.1956E-04	0
26	Mean	0.3425190851	3.3580	0	0	9.2611E-04	0
	S.D	0.2793844328	1.7072	0	0	4.9399E-04	0
	Best	0.107854422	5068.484	0	0	2.1198E-07	0
34	Mean	0.190997743	5661.874	0	0	2.2829E-06	0
	S.D	0.140423693	445.5457	0	0	2.6432E-06	0
	Best	1.1209E+10	3.8475	1.338E+06	0.8768	1.668E+06	1.1681
41	Mean	1.1944E+10	4.6841	6.633E+06	0.9015	3.489E+06	1.17
	S.D	8.1539E+08	0.9884	4.860E+06	0.0161	1.179E+06	0.0268
	Best	2.4512E+10	79.3684	3.454E+07	94.4521	5.634E+07	98.8592
42	Mean	2.4700E+10	84.9612	7.467E+07	95.4800	7.775E+07	98.8714
	S.D	1.6453E+10	6.9411	3.081E+07	0.4430	1.162E+07	0.00124
	Best	2285.5368	668.9409	620.068	1.9610E-14	678.622	0
52	Mean	2321.4304	719.3644	668.383	1.5370E-05	732.821	0
	S.D	31.2049	30.0033	38.864	5.8759E-05	21.956	0
	Best	0.010911	6346.161	0	0	2.4143E-07	0
60	Mean	0.036556	6421.448	0	0	0.1234	0
	S.D	0.03629	89.0268	0	0	0.2983	0

Table 5.9 Results for Large-scale problems, D = 2000

f_n	Statistic	PSO2011	ABC	FPA	GWO	FFA	NPO
	Best	2.5165E+23	24.9891	187857	1.8118E-06	272221	0
3	Mean	6.4323E+24	29.1845	334987	3.81559	415776	0
	S.D	9.8032E+14	3.8541	939815	14.55623	68451	0
	Best	1.338E+24	35572.845	5.8022E+23	1.1272E+04	7.261E+24	1.054E-3
14	Mean	1.562E+24	35821.841	1.5932E+24	1.3308E+04	9.533E+24	0.00784
	S.D	1.963E+23	227.2855	6.4922E+23	1.3890E+03	1.091E+24	0.01474
	Best	0.100132	1.6976	7.8722E-11	6.1658E-07	1.1973E-07	0
15	Mean	0.229218	2.0018	2.7013E-06	0.00436	3.8691E-06	0
	S.D	0.128811	0.2633	9.2661E-06	0.019509	5.1145E-06	0
	Best	3.956E+05	12.6052	2383	0.0037	5800	2.3419E-07
16	Mean	4.086E+05	16.1874	4665	0.0083	6941	2.2757E-05
	S.D	1.167E+05	2.5176	1393	0.0024	506.93	2.0576E-05
	Best	73063.711	57632.748	16622	1.9437E-10	22988	0
17	Mean	73635.765	60089.174	19997	2.6938E-10	23692	0
	S.D	763.962	2276.845	16672	4.9538E-11	358.70	0
	Best	96.0583	99.6208	37.7266	75.747423	97.8796	0
18	Mean	96.8142	99.6793	43.4886	81.18971	98.3709	0
	S.D	0.7548	0.0413	3.0758	3.262956	0.2807	0
	Best	-	22.1478	INF	7.1622E+03	INF	0
19	Mean	-	25.9658	INF	7.4488E+03	INF	0
	S.D	-	1.4414	NaN	1.7400E+02	NaN	0
	Best	3.0892E+11	4.472E+11	2366245	7.8008E-38	1.9167E+07	0
20	Mean	3.2103E+11	5.334E+11	1983923	1.7615E-04	3.0004E+07	0
	S.D	1.1225E+10	5.517E+10	1472984	9.6453E-04	5.624E+06	0
	Best	3.7959E+06	3.0568E+06	23426	2.7822E-19	4.3745+07	0
21	Mean	3.8229E+06	3.1489E+06	33365	5.6527E-19	4.3899+07	0
	S.D	4.4358E+04	9.1274E+04	48586	2.2700E-19	1.0478+04	0
	Best	71481	104.4456	15810	0	21986	0
22	Mean	72484	112.6341	19156	0	22518	0
	S.D	887.383	29.2245	18452	0	287.61	0
	Best	3798199	142.2355	240181	0	424709	0
23	Mean	3.815E+06	162.2411	327451	0.0333	448213	0
	S.D	18168.498	56.5468	482315	0.1825	15294	0
	Best	3.7712E+07	1.1104	265134	7.0539E-19	3888146	0
25	Mean	3.8317E+07	1.3347	322812	4.5726E-18	4221967	0
	S.D	5.1128E+05	0.0114	384351	9.7734E-18	1388215	0
	Best	0.03245	2.3939	0	0	1.1085E-04	0
26	Mean	0.26227	3.0277	0	0	9.8684E-04	0
	S.D	0.20556	0.4295	0	0	5.2224E-04	0
	Best	0.17971	2.741E+04	0	0	3.8677E-07	0
34	Mean	0.26673	2.818E+04	0	0	0.0018	0
	S.D	0.09525	483.4293	0	0	0.0100	0
	Best	2.7797E+10	2.8421	4794010.6	0.9801	1.634E+07	1.6625
41	Mean	2.4371E+10	2.9982	1568531.7	1.0044	2.364E+07	1.7618
	S.D	7.3089E+10	0.0854	8206735.6	0.0121	5.615E+06	0.00426
	Best	5.983E+10	201.948	4.3962E+07	194.3941	2.181E+07	98.8203
42	Mean	5.049E+10	206.665	1.7392E+08	195.402	2.796E+08	98.3888
	S.D	8.779E+08	0.68451	1.0067E+08	0.514705	2.810E+07	0.04481
	Best	4665.602	2695.774	1357	1.3989E-11	1558	0
52	Mean	4718.114	2774.461	1422	2.3068E-04	1604	0
	S.D	50.057	67.6205	76.7146	4.7534E-04	25.48	0
	Best	3.7212E-04	20592.204	0	0	7.9327E-07	0
60	Mean	0.3368912	21365.648	0	0	0.0029	0
	S.D	0.03807503	489.8769	0	0	0.0156	0

5.4 Convergence Analysis

The convergence curves for several test functions of NPO and the other algorithms are provided in Figures 5.4A – Figure5.4F for the first 100 iterations. It is clear that NPO has fast convergence as compared with the other algorithms, because of two reasons. Firstly, the semi-circular distribution and the leadership transition change the position of the families (solutions) faster than the other algorithms, in other words, these two steps enhance the local search mechanism and get a new local best solution each iteration. At the same time, when the families are distributed based on semi-circular distribution, their new positions depend on their sheikhs (local best solutions), meaning that the families are converging fast enough toward the optimal solution. Secondly, the periodical meeting (meeting room approach) increases the exploration ability of NPO by sharing the information between the Sheikhs (local best solutions), which enhances the searching ability of the families when they are looking for new positions.

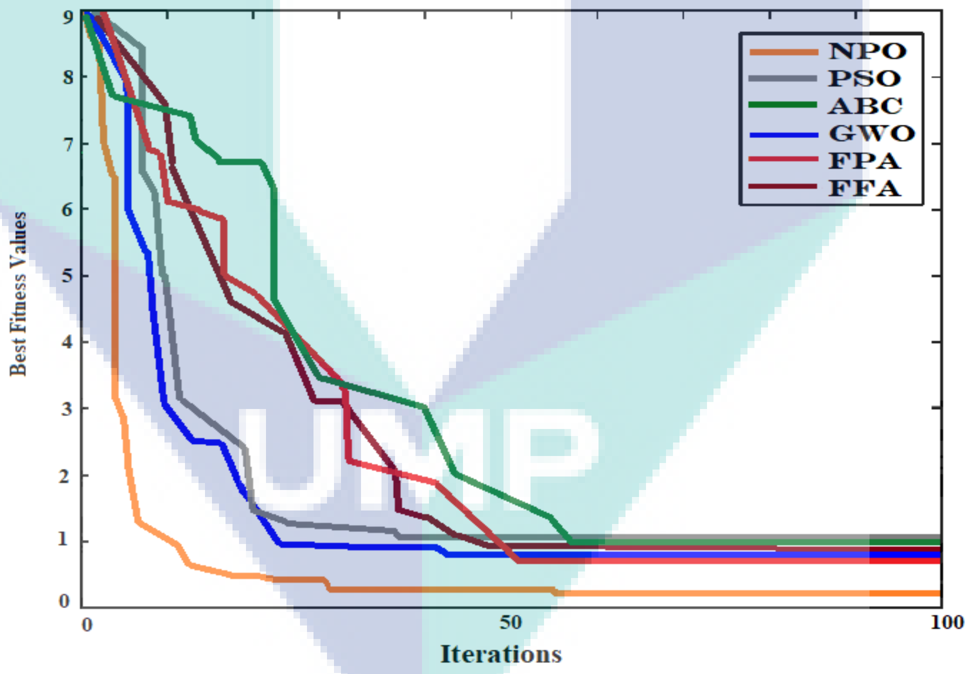


Figure 5.5A Convergence curve for f_7

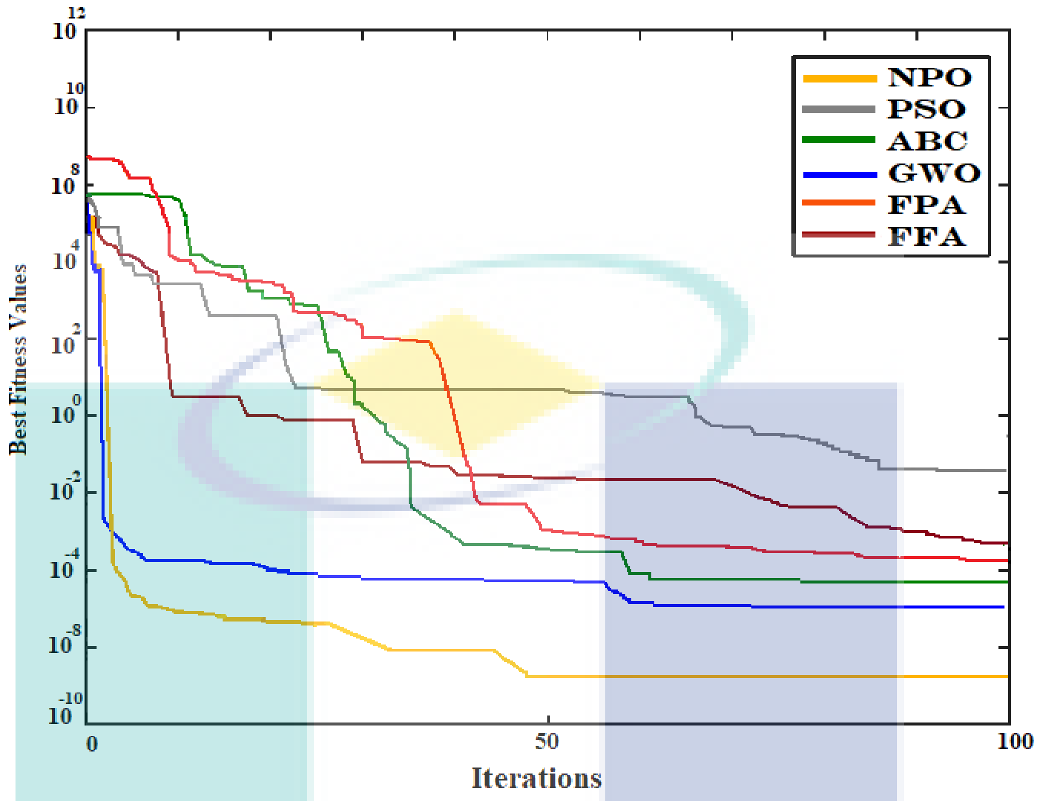


Figure 5.5B Convergence curve for f_{16}

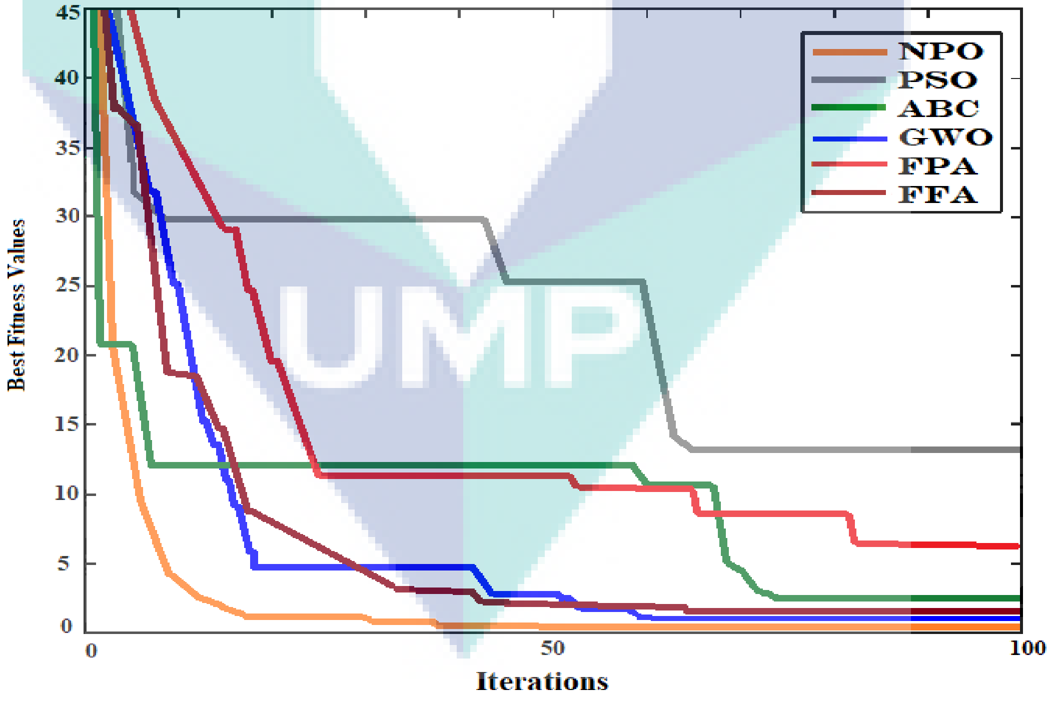


Figure 5.5C Convergence curve for f_{21}

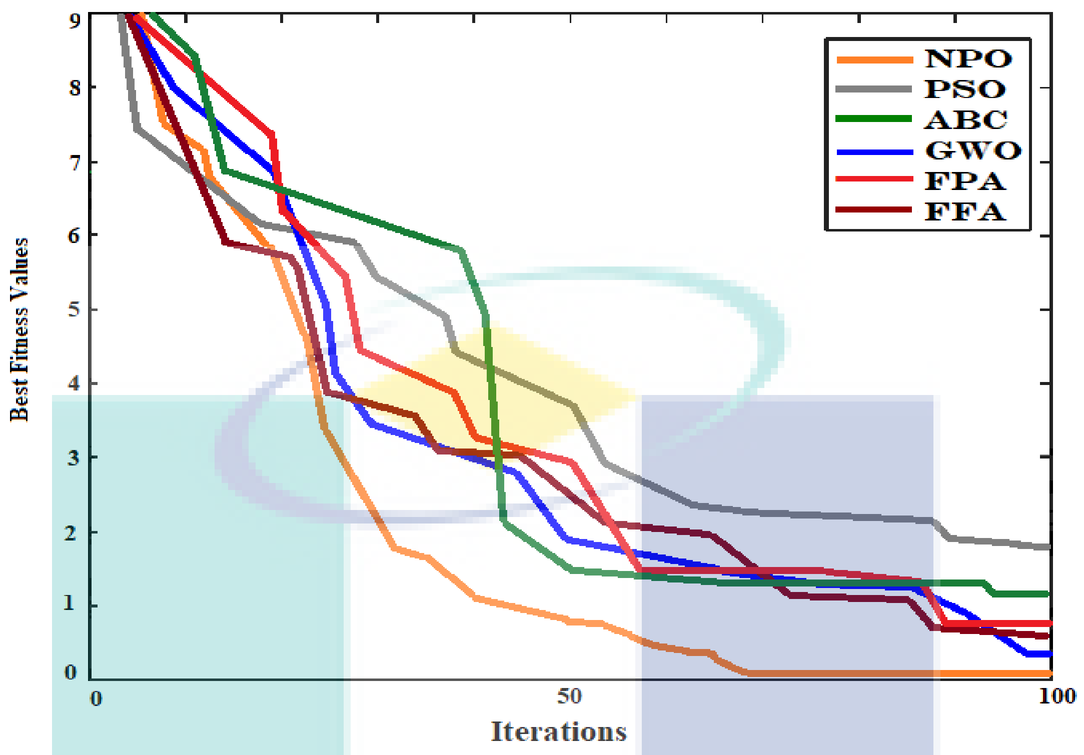


Figure 5.5D Convergence curve for f_{26}

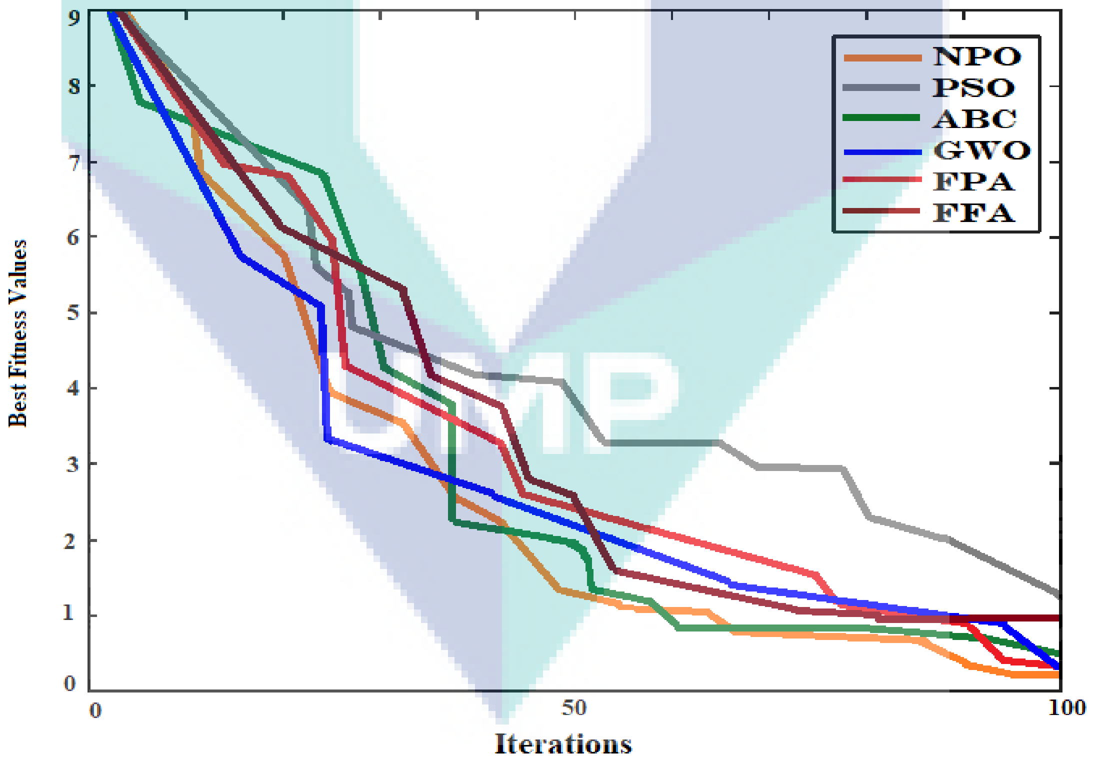


Figure 5.5E Convergence curve for f_{34}

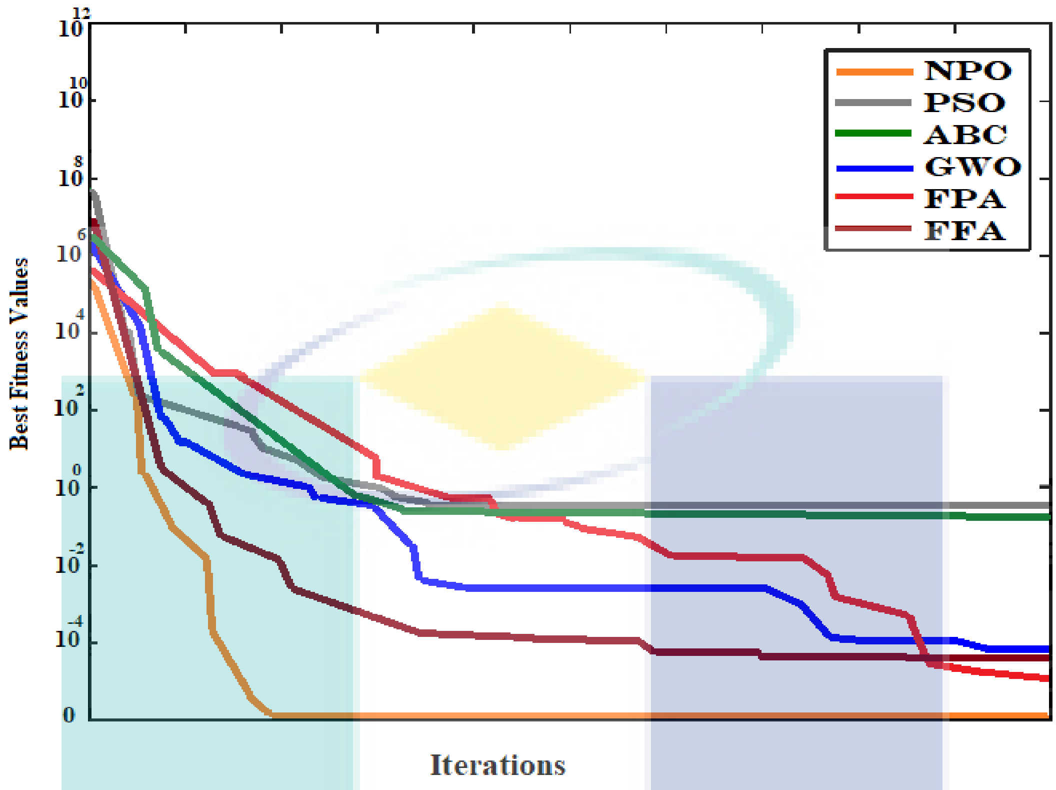


Figure 5.5F Convergence curve for f_{60}

Figure 5.5 Convergence curves for functions($f_7, f_{16}, f_{21}, f_{26}, f_{34}, f_{60}$)

As a summary of the convergence, each member (family) in NPO has its own responsibility to improve its position within the clan, and help the clan to find better position within the desert. This is evident in the convergence curves, NPO showed a superior performance as compared to the rest algorithms.

5.5 Exploitation and Exploration Analysis

To analyse the two highly influential factors (exploration and exploitation) of the metaheuristics, five commonly used numerical optimization problems with different modality were employed with 30 dimensions. These test function are ($f_{19}, f_{21}, f_{26}, f_{41}, f_{60}$). This section focuses mainly on calculating diversity in swarm during iterations instead of running the algorithm over certain number of independent runs and averaging the results. Accordingly, we executed algorithm once, as our preliminary experiments also evidenced insignificant difference in results over multiple runs.

The exploitation has been calculated by using a counter in three different parts, first part when all families are initialized around the Sheikh's tent. While the second and third parts when each time the Leadership Transition is executed. On the other hand, the exploration has been calculated by using the same method, in three different parts as well. First part when the Sheikh's are initialized in the initial meeting, while the second part during the families searching. Final part, when any Sheikh update his own position inside the MRA. Table 5.10 presents the best objective function value found by NPO and the other metaheuristics, exploration, and exploitation.

Table 5.10 Results of Exploration and Exploitation

f_n	Measurements	PSO	ABC	FPA	GWO	FA	NPO
f_{19}	Solution	1.7985	1.30E-14	47493.2929	8.28E-41	0.27895	0
	Exploration	35%	58%	85%	32%	83%	22%
	Exploitation	65%	42%	15%	68%	17%	78%
f_{21}	Solution	1.2945	0	2.66E-52	0	0.0012864	0
	Exploration	33%	59%	63%	68%	88%	46%
	Exploitation	67%	41%	37%	32%	12%	54%
f_{26}	Solution	1.2293	0.020580523	0	0	2.86E-05	0
	Exploration	36%	61%	60%	45%	92%	78%
	Exploitation	67%	39%	40%	55%	08%	22%
f_{41}	Solution	8.8242	3.82E-16	1.3124	0.0065555	1.20E-05	0.008915
	Exploration	40%	58%	59%	71%	86%	87%
	Exploitation	60%	42%	41%	39%	14%	13%
f_{60}	Solution	0.024484	5.68E-14	0	0	4.94E-10	0
	Exploration	56%	72%	74%	76%	82%	71%
	Exploitation	44%	28%	26%	24%	18%	29%

From Table 5.10, there is a dynamic problem nature-related difference between the exploration and exploitation features of NPO. The flexibility of this difference could be attributed to two reasons: the first reason is the enhancing effect of MRA on the searching process through guiding the normal Sheikhs towards better positions once established; the second reason is related to the process of checking for any family with a better fitness than the Sheikh of the clan (Step 10 in the pseudocode). This condition controls the algorithm and decides whether a local search (leadership transition and semi-circular distribution – or exploitation) or a global search (families searching – or exploration) is needed. Unlike the other algorithms, the exploitation and exploration functions are simultaneously executed in the NPO using different governing equations.

5.6 Discussion

This section discusses the outcomes of NPO, and attempts to answer the following question: ‘why is NPO efficient?’ In fact, two primary reasons can be outlined in this case, which are: 1) NPO has good exploration and exploitation capabilities, and 2) NPO has a powerful mechanism that balances exploration and exploitation capabilities. The exploration is applied twice, the first time occurs when the leaders are initialized at the first meeting and meet at the periodical meeting, while second time takes place when the families are searching in the search space, or in precise, the third step. The exploration of NPO differs from that of other metaheuristics, in which it explores the search space by employing several members of swarms, while other swarm-based algorithms commonly use a specific mechanism between the global best solution and the whole swarm. Moreover, the meeting room approach (MRA), which is proposed in this paper, forces the normal leaders to follow the best leader by using the direction variable Ψ. This variable guides them towards better places, in precise, they may find better positions for their clans. In this paper, two values were employed for the direction variable, +1 or -1, because the values of fitness appeared either positive or negative, while in future studies, the researcher may use varied values based on their case studies, if these values do not suit them.

The exploration ability of NPO was optimum when NPO was applied on multimodal test functions, which comprised of 35 test functions. NPO successfully discovered 29 optimal solutions. In addition, NPO proved that it possesses the ability to avoid all local optima and could approach the global optima on most of these tests. The convergent curves showed that NPO had the fastest convergence on multimodal test functions as well.

On the other hand, the exploitation stage consists of two steps: semi-circular distribution, and leadership transition. These two steps represent the local search mechanism of NPO, while the families searching explore the search space solutions generated by the other two operators. Besides, it is worth to mention that each clan with these three operators reflects an independent search algorithm, which indicates that search algorithms are embedded in NPO $\#Clans$ (no. of clans). For each iteration (generation) in NPO, the families in each clan search for better places to move to, thus discovering sheikhs, and the clans can be enhanced internally. Even if those sheikhs fail to emerge as better than the global best sheikh, they still represent an enhancement in NPO, thus leading to an enhancement in the searching process, when MRA is applied. Figure 5.6 illustrates an idea of the processes that take place in both stages and the general block diagram of NPO.

It is obvious that NPO does not contain any controlling parameters, except for structural parameters, i.e., number of clans and number of iterations. Although these parameters do not influence the search behaviour of NPO, they do have an impact on the probability of finding the best solutions, or in precise, more families or more clans find the solutions faster in the expense of execution time. This study had examined five clans and ten families in each clan (50 swarm size in total), in which the performance was found efficient in terms of execution time.

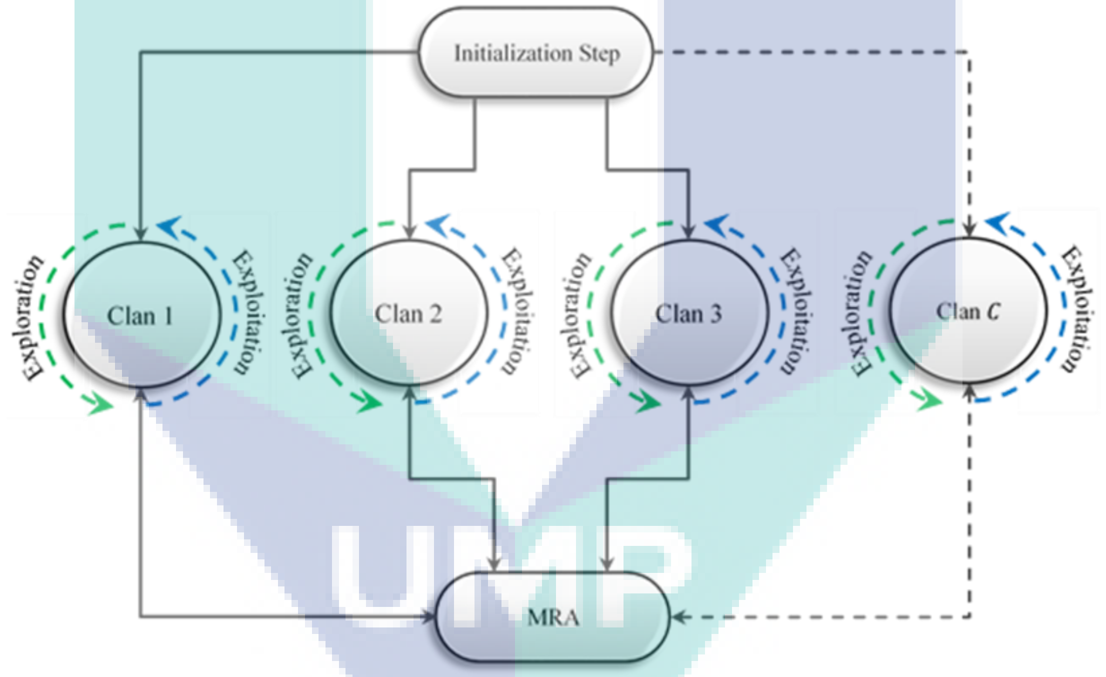


Figure 5.6 Exploration and exploitation of NPO

It is important to note that NPO exhibited exceptional performance with noise test function, especially Quartic test function (f_{16}). To the best of our knowledge, no algorithm in the literature has recorded the performance level achieved in this study. On the other hand, some functions, such as Matyas (f_6) and Stepint (f_{24}), proved to be difficult functions since the flatness of the function did not provide the algorithm any information to channel the search space towards the best solutions. NPO, nonetheless, attained the best solution for these two functions, hence proving its efficacy in solving problems with limited information.

With proven efficiency of NPO, it also has some shortcomings that should be investigated in future studies. On drawback of NPO is its failure in solving several test functions, especially those in the form of multimodal, such as $(f_{41} - f_{44}, f_{47})$. Next, NPO failed in seeking the best solution for f_{13}, which refers to a unimodal test function in the used number of iterations, where it started rapidly at the beginning of the search, but upon reaching the fitness value (209) that is close to the optimal solution (210), the convergence became very slow. Nevertheless, NPO attained the optimal solution for this test when 5,000 iterations were used. Figure 5.7portrays the convergence of NPO for f_{13}.

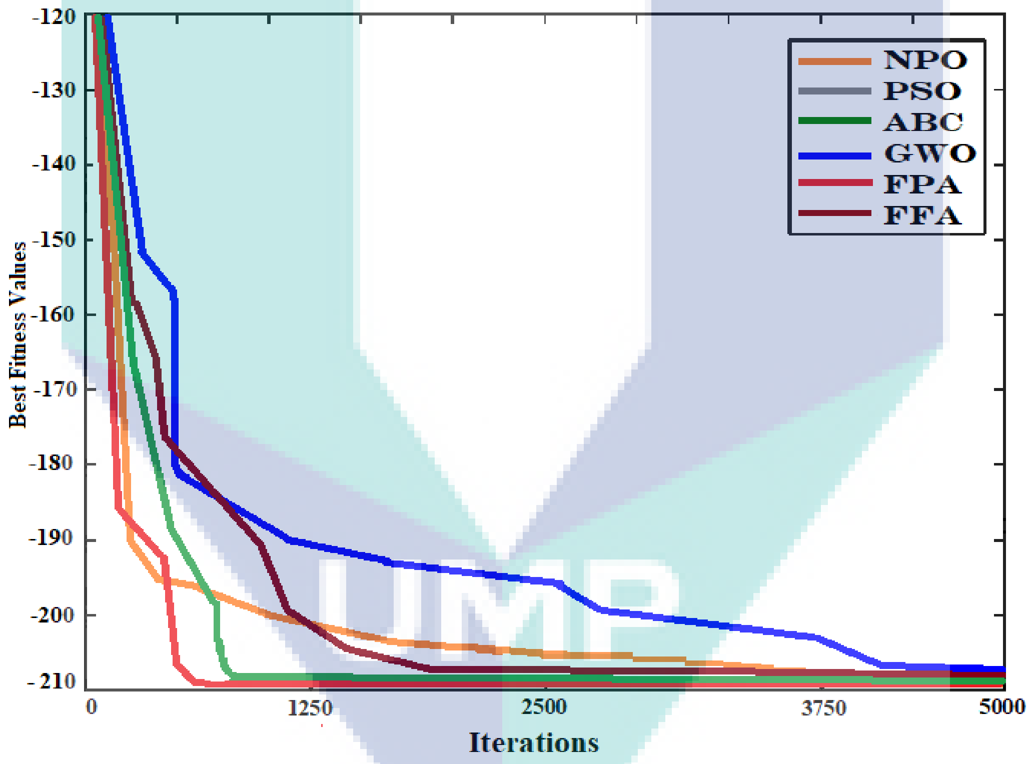


Figure 5.7 Convergence curve for f_{13}

As mentioned previously, NPO and the other algorithms are swarm-based metaheuristics. Hence, they have been evaluated within the same environment. A time-based comparison, however, showed that NPO reached the optimal solutions for most of the tests within shorter period of time, in comparison to other algorithms. Figure 5.8 displays the time-based comparison of single run for each test. Figure 5.8 shows that the

NPO has a superior performance in terms of execution time in many test functions, such as ($f_1, f_4, f_6 - f_{11}, f_{13} - f_{32}, f_{34}, f_{38}, f_{39}, f_{48}, f_{49}, f_{52}, f_{53}, f_{56}, f_{60}$. Meanwhile, the performance of NPO appeared moderate for the other test functions.

Almost all metaheuristic algorithms are simple in terms of complexity, and thus they are easy to implement. The time complexity of the NPO algorithm is compared with the other metaheuristics in Table 5.11.

Table 5.11 Comparison of time complexity

Algorithm	**Time Complexity**	**Rank**
PSO	$O(T.P)$	2
ABC	$O(T.(E+O).A)$	3
FPA	$O(T.N)$	2
GWO	$O(T.G)$	2
FFA	$O(T.N^2)$	4
NPO	$\boldsymbol{O(T.C.(F-1))}$	1

From Table 5.11, it is obvious that the complexity of the algorithms is almost the same. However, NPO requires less amount of time due to two main reasons. First, NPO contains a condition which controls the searching process, local search (Semi-Circular Distribution Operator) or global search (Families Searching Operator). Meaning that, NPO does not require to iterate all families for exploration or exploitation inside each iteration. Second, NPO deals with several swarms, not all these swarms' members are moving – or searching – because the leader stays at his own positions.

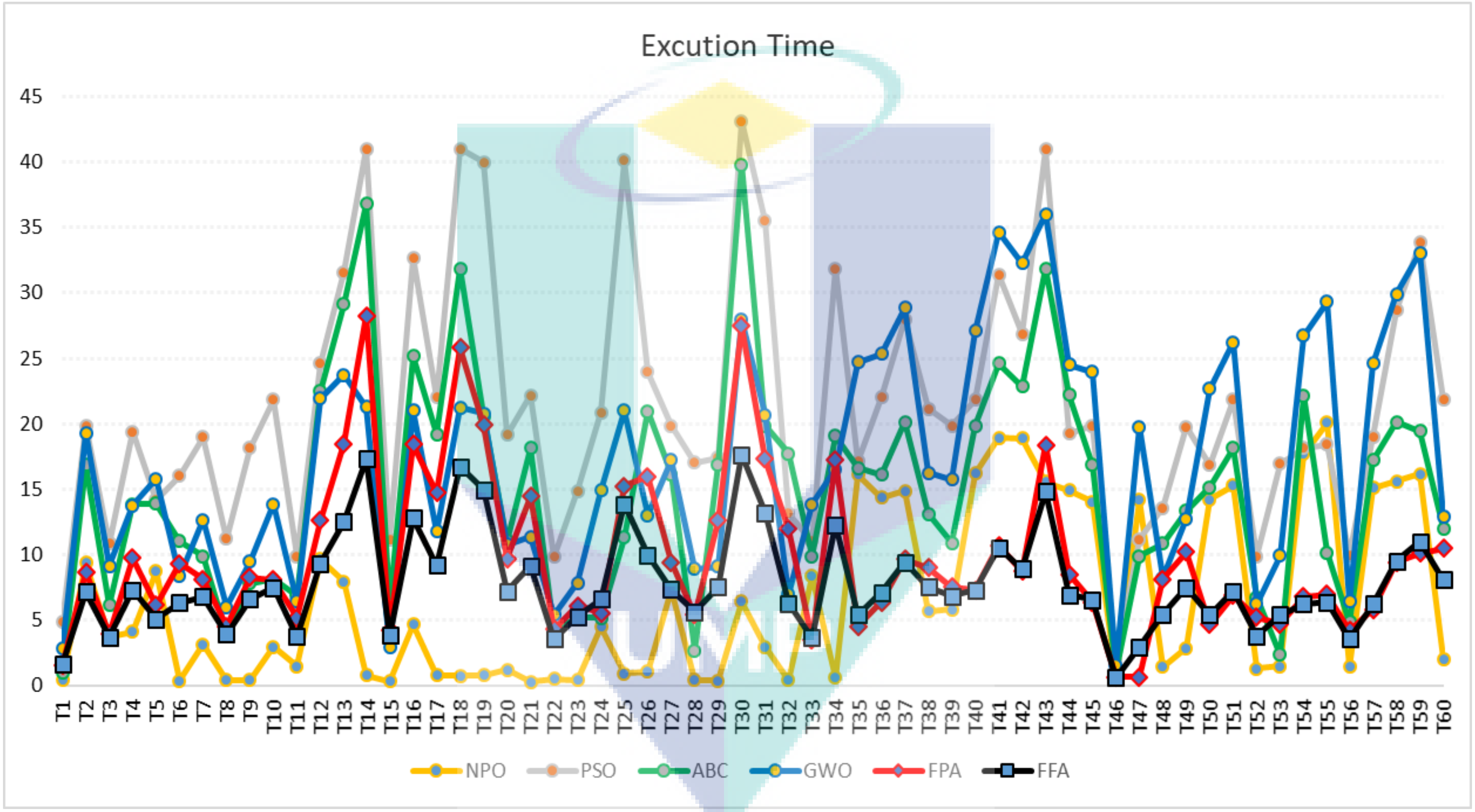


Figure 5.8 Time-based comparison between all metaheuristics

The Meeting Room Approach (MRA) was individually evaluated by developing a new version of Particle Swarm Optimization algorithm which is called "MPSO" based on the proposed MRA. The performance of the developed MPSO was compared to the standard PSO and a "Master-Slave PSO model". In general, the results showed a better performance of MPSO compared to the other algorithms on the same test functions. The results also showed that MPSO can solve both unimodal and multimodal test functions faster than the standard PSO algorithm. MPSO is explained and discussed in details in Appendix B.

5.7 Summary

The results obtained by NPO over a new combination set of sixty benchmark test functions has been analyzed and discussed in this chapter. It also shows the ability of NPO for solving the large-scale optimization problems. The new proposed combination of test functions were classified into four different classes, Unimodal Separable (U-S), Unimodal None Separable (U-N), Multimodal Separable (M-S), and Multimodal None Separable (M-N).

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Introduction

Several metaheuristic algorithms and improvements to the existing ones have been presented over the years. Most of these algorithms were inspired either by nature or the behaviour of certain swarms, such as birds, ants, bees, or even bats. In this thesis, a new nature inspired metaheuristic have been proposed for global optimization problems. This chapter presents a summary of the research, contributions and description of the future direction of the research as presented in this thesis.

6.2 Research Summary

Hard optimization problems are roughly defined as problems that are difficult to be optimally solved using any deterministic method within a ‘‘reasonable’’ time frame. Such problems are classified based on their nature into several categories, such as constrained or unconstrained, continuous or discrete, static or dynamic, mono or multi-objective. These problems are satisfactorily solved using metaheuristics. Metaheuristic are algorithm which have the capability of solving a wide range of hard optimization problems without necessarily adapting to each problem. The prefix ‘‘meta’’ (Greek prefix) in the word metaheuristic indicates advanced nature of these heuristics compared to the problem-specific heuristics.

Problems that cannot be satisfactorily solved with the problem-specific algorithms are usually solved with metaheuristics, especially the complex problems encountered in the industries and service areas such as finance, engineering, and production management. Virtually all metaheuristics are nature-inspired (based on some physical, biological, and ethological principles), do not use the objective function gradient or Hessian matrix, use stochastic components (involving random variables), have several parameters that demands fitting into the problem to be solved. Metaheuristics have received a great interest in the last 3 decades.

The success of a metaheuristic in solving a given problem is determined by its ability to strike a balance between exploration (diversification) and exploitation (intensification). The regions of the search space with high quality solutions are identified through exploitation. Furthermore, exploitation is important for intensifying the solution search to some promising areas of the accumulated search experience. The existing metaheuristics differ from each other in the way they strive to strike this balance. Metaheuristics can be classified based on many classification criteria. These criteria may include a consideration of their features with respect to the search path, the memory requirement, the kind of neighbourhood exploration deployed, or the number of current solutions carried from one round of iteration to the other. In the literature, metaheuristics are mainly classified as either single solution-based (SSB) or population-based (PB) metaheuristics. The basic SSB metaheuristics are mainly more exploitation oriented while the basic PB metaheuristics are more exploration oriented.

There are several issues faced by the metaheuristics, and one of such issues is balancing exploration with exploitation. This issue has been previously discussed. Algorithmic search processes are delayed when the exploratory capability of the algorithm is more than its exploitation capability; but premature convergence is experienced when the reverse is the case. Hence, there is a need to ensure a balance between these two algorithmic searching behaviours for an optimum searching process. Another problem of metaheuristics is their dependence and use of some control parameters whose values significantly controls algorithmic searching process. Hence, there is a need to tune these parameters for a better algorithmic performance.

This study strives to provide answer to one basic question; which is: “Why the need to design new metaheuristics?” This is the most important question to justify the need for new metaheuristics when there are currently tens nature-inspired metaheuristics with unique designs and good performances in solving several engineering and industrial problems. However, the emergence of new sources of inspiration directly translate into the development of novel metaheuristics. In general, this question has been addressed in chapter two but the answers can be summarized thus:

1. The optimal solution to most engineering and industrial optimization problems is difficult to find due to their nonlinearity (NP-hard) under stringent constraints.

2. It is a fact that no single optimization method can outperform all other methods on all problems, but there several optimization methods which are dedicated to finding solution to different types of problems. Similarly, some algorithms are developed to provide good solutions to different types of problems, but may be outperformed by the highly specialized methods in some situations. Wolpert et al. (1997) formalized this fact in their *No Free Lunch Theorem* (NFL) for search and optimization frameworks, where they specified on single-objective optimization and proved that, for all optimization algorithms, the sum of the values of any performance measure (such as the objective value of the best solution candidate discovered until a time step **m**) over all possible objective functions **f** is always similar.

3. Some of the drawbacks of the existing metaheuristics have been discussed in chapter two. At first, the searching capability of all metaheuristics depend on two main components which are exploration and exploitation, and these two components must be optimally balanced (this is still a major problem). Secondly, the searching ability of most metaheuristics are governed by more than one control parameters whose optimal values are difficult to be determined. This is the justification for the need to design new parameters-free metaheuristics.

Three main objectives were considered in this study; the first objective was to propose a novel parameter-free multi-swarm metaheuristic inspired by the movement of nomads when searching for the sources of food in the desert, while the second objective was to implement the proposed algorithm for solving the unconstrained continuous global optimization problems. The final objective was to evaluate the proposed algorithm based on a combination set of benchmark test functions.

To address the first objective, a novel parameter-free nature-inspired metaheuristic for solving global optimization problems was designed. The proposed metaheuristic was inspired by the movement of nomads in the desert when searching for the sources of food. The proposed algorithm (called Nomadic People Optimizer (NPO) algorithm) depends on a new multi-swarm cooperative approach called “Meeting Room Approach (MRA)” for establishing the best solutions, hence, a multi-swarm metaheuristic.

The NPO consists of two types of families; the first family is the Sheikh’s family which has the best position and fitness value (positioned next to the food source). The second family is the normal families. Additionally, the NPO consists of five operators; the first operator which is responsible for initiating the Sheikh’s families only is called “Initialization”. This implies that the Sheikhs’ families are initialized at the first stage, and each one represents a clan. The second operator is the semi-circular distribution of the normal families around the Sheiks tent. This operator is responsible for distributing or initializing the normal families around the Sheikh’s family. The fourth operator is the leadership transition which is responsible for transferring leadership to another family with more power (may be a new source of food). These two operators represent the exploitation part of the NPO.

The third operator is families searching. This operator is executed when the normal families have a lower fitness value compared to that of the Sheikh’s family. The normal families search for better places by updating their positions. When a family finds a better new position (i.e. a better fitness value), it becomes the new Sheikh. This operator represents the exploration part of the NPO. The final operator is the periodical meeting (i.e. Meeting Room Approach “MRA”), which is responsible for balancing between the exploration and exploitation components of the NPO.

Meeting Room Approach “MRA” is a general multi-swarm approach, which consists of several swarms called ‘clans’; each clan has a leader which represents the best solution in the clan (the local best). All the clan leaders periodically meet to select the best leader among them (the global best). The best leader has control over all the clan leaders with respect to reaching the best positions. The interaction between the best leader (global best) and the normal clan leaders (local best) influence the balance between the exploration and exploitation capabilities and maintain a suitable diversity in the population even when approaching the global solution; thus, reducing the risk of local sub-optima convergence.

Regarding the third objective, a new combination of unconstrained test functions which contain 60 test functions was proposed for the evaluation purposes. These test functions were divided into four groups (Unimodal Non-Separable (U-N) with 14 tests, Unimodal Separable (U-S) with 11 tests, Multimodal Non-Separable (M-N) with 26 tests, and Multimodal Separable (M-S) with 9 tests). In addition, this set can be divided into two groups, fixed and dynamic dimensions. There are 18 tests were used with 30 dimensions, meaning that these tests are dynamic; while the other tests are fixed dimensions. These 18 tests have been used to evaluate the scalability of the proposed algorithm when solving the large-scale problems.

6.3 Conclusion

The major contribution of this study was the design and implementation of a novel nature-inspired metaheuristic for unconstrained (normal and large-scale) optimization problems. The proposed NPO depends on a new multi-swarm approach and this is another contribution of this study. The conclusions drawn from this study are summarized as follows:

i) NPO algorithm has a unique and simple structure as it was inspired by a human life behavior (nomadic life of the nomads). Additionally, NPO does not require any controlling parameter as it requires only two structural parameters which are the number of clans or Sheikhs and the number of families.

ii) The NPO algorithm was evaluated over sixty benchmark test functions in both unimodal and multimodal forms. The proposed algorithm was able to solve 52 problems (23 unimodal and 29 multimodal functions).

iii) The scalability of NPO algorithm was evaluated over eighteen large-scale benchmark test functions with four variable sizes (100, 500, 1000, and 2000). The NPO had a superior performance over sixteen test functions, which proved its ability to solve problems with large number of decision variables.

iv) The NPO showed a very good performance with noise problems, such as Quartic test function. The NPO achieved results that had never been reported for any other algorithm in the current literature.

v) The NPO achieved good results when solving problems with limited information, such as Matyas and Stepint test functions. These problems do not provide any information on how to channel a search process toward the best solutions due to the flatness of the functions.

vi) The convergence curves showed that NPO had the fastest convergence on multimodal test functions. The NPO proved its ability to avoid all local optima and reach the global optima on most of the examined test function. This suggests that the NPO had a good exploration (families searching operator) and balancing (meeting room approach) mechanism.

vii) The MRA was proven as a good balancing mechanism when applied to PSO algorithm.

6.4 Suggestions for Future Works

In this study, a novel nature-inspired metaheuristic was for solving unconstrained (normal and large-scale) optimization problems. Although the proposed metaheuristic has been successfully applied to several benchmark test functions with promising results, there are several suggestions for future investigations. These suggestions are categorized into two classes as follows:

6.4.1 Applications

The NPO algorithm can be used to solve different case studies, such as:

i) NPO can be applied in the Machine Learning field such as in the training of Artificial Neural Networks (ANNs). The NPO can be used as ANN training algorithm for classification and regression problems. In other words, the NPO searches for the best weights and biases for the feedforward neural networks which enhances the classification accuracy of the model or decreases the prediction error with the forecasting case studies.

ii) NPO can be used for selecting the best subset features; meaning that it can be applied as a wrapper feature selection algorithm. More experimentation and investigation may be required to observe the behavior of NPO in such problems.

iii) NPO can be applied for solving combinatorial optimization problems such as Traveling Sales Man problem or t-way test problems.

iv) Several scheduling problems can be investigated and solved using NPO, such Time table problem or CPU task scheduling problem.

v) The proposed multi-swarm approach (MRA) can be hybridized with other nature-inspired swarm-based metaheuristics such as Firefly algorithm, Bat algorithm, and Grey wolf optimizer algorithm for enhancing their exploration and exploitation capabilities.

6.4.2 Modifications

In this study, the standard version of NPO was proposed and examined; however, there are several modifications to this version which can be considered for future studies, such as:

i) The standard version of the NPO is a single objective algorithm, meaning that the objective function deals with minimizing or maximizing only one objective. Therefore, this version can be modified to handle more than one objective, implying the development of a multi-objective version of the NPO algorithm.

ii) An evolutionary version of the NPO can be proposed by applying a mutation operator to the fourth operator (i.e. Leadership Transition), especially with the combinatorial optimization problems. This operator may enhances the exploitation capability of NPO.

iii) In the real life of nomads, some families can migrate from a clan to another clan closer to food sources; therefore, a new evolutionary version of NPO can be proposed with the migration operator. This operator may enhance the exploration capability of NPO.

iv) The first operator (i.e. Initialization) utilized a uniform distribution equation for generating random positions for the Sheikhs; however, there is a chance that this equation may initialize the Sheikhs at positions far from the optimal solution. Therefore, this operator may be enhanced by applying another randomization equation such as chaotic maps.

v) The best Sheikh in the meeting room does not move to any new position after sharing information with the normal Sheikhs; therefore, this part may be enhanced by applying a kind of movement to the best Sheikh which may lead to better exploration, or at least, can decrease the chances of falling in the local optima.

REFERENCES

Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic algorithms a comprehensive review. In *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (pp. 185–231). Elsevier. https://doi.org/10.1016/C2016-0-03919-0

Agarwal, P., & Mehta, S. (2014). Nature-inspired algorithms: state-of-art, problems and prospects. *International Journal of Computer Applications*, *100*(14), 14–21. https://doi.org/10.5120/17593-8331

Ahmadi-Javid, A. (2011). Anarchic Society Optimization: A human-inspired method. In *Proceddings of IEEE Congress of Evolutionary Computation (CEC 2011)* (pp. 2586–2592). https://doi.org/10.1109/CEC.2011.5949940

Akay, B., & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-010-0393-4

Al-Dabbagh, R. D., Neri, F., Idris, N., & Baba, M. S. (2018). Algorithmic design issues in adaptive differential evolution schemes: review and taxonomy. *Swarm and Evolutionary Computation*. https://doi.org/10.1016/J.SWEVO.2018.03.008

Alshamlan, H. M. (2018). Co-ABC: Correlation artificial bee colony algorithm for biomarker gene discovery using gene expression profile. *Saudi Journal of Biological Sciences*. https://doi.org/10.1016/j.sjbs.2017.12.012

Alshamlan, H. M., Badr, G. H., & Alohali, Y. A. (2015). Genetic bee colony (GBC) algorithm: a new gene selection method for microarray cancer classification. *Computational Biology and Chemistry*. https://doi.org/10.1016/j.compbiolchem.2015.03.001

Andrei, N. (2008). An unconstrained optimization test functions collection. *Advanced Modelling and Optimization*, *10*(1), 147–161.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In *Proceedings of IEEE Congress on Evolutionary Computation, (CEC 2007)* (pp. 4661–4667). https://doi.org/10.1109/CEC.2007.4425083

Aydilek, İ. B. (2018). A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems. *Applied Soft Computing Journal*, *66*, 232–249. https://doi.org/10.1016/j.asoc.2018.02.025

Azrag, M. A. K., Kadir, T. A. A., Odili, J. B., & Essam, M. H. A. (2017). A global african buffalo optimization. *International Journal of Software Engineering and Computer Systems*, *3*(3), 138–145.

Barták, R., Salido, M. A., & Rossi, F. (2010). Constraint satisfaction techniques in planning and scheduling. *Journal of Intelligent Manufacturing*, *21*(1), 5–15. https://doi.org/10.1007/s10845-008-0203-4

Beni, G., & Wang, J. (1991). Theoretical problems for the realization of distributed robotic systems. In *Proceedings of International Conference on Robotics and Automation* (pp. 1914–1920). Sacramento, CA, USA,: IEEE.

Bénichou, O., Loverdo, C., Moreau, M., & Voituriez, R. (2006). Two-dimensional intermittent search processes: an alternative to lévy flight strategies. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, *74*(2). https://doi.org/10.1103/PhysRevE.74.020102

Bénichou, O., Loverdo, C., Moreau, M., & Voituriez, R. (2011). Intermittent search strategies. *Reviews of Modern Physics*, *83*(1), 81–129. https://doi.org/10.1103/RevModPhys.83.81

Benkercha, R., Moulahoum, S., & Kabache, N. (2017). Combination of artificial neural network and flower pollination algorithm to model fuzzy logic MPPT Controller for Photovoltaic systems. In *Proceedings of 18th International Symposium on Electromagnetic Fields in Mechatronics, Electrical and Electronic Engineering (ISEF)* (pp. 1–2). IEEE.

Beyer, H. G., Finck, S., & Breuer, T. (2014). Evolution on trees: on the design of an evolution strategy for scenario-based multi-period portfolio optimization under transaction costs. *Swarm and Evolutionary Computation*, *17*, 74–87. https://doi.org/10.1016/j.swevo.2014.03.002

Birattari, M., Paquete, L., St, T., & Varrentrapp, K. (2001). Classification of metaheuristics and design of experiments for the analysis of components. *Technology*.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, *35*(3), 189–213. https://doi.org/10.1007/s10479-005-3971-7

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. *Journal of Artificial Societies and Social Simulation* (Vol. 4). https://doi.org/10.1080/09540090210144948

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*, 82–117. https://doi.org/10.1016/j.ins.2013.02.041

Caraffini, F., Neri, F., & Iacca, G. (2017). Large scale problems in practice: the effect of dimensionality on the interaction among variables. In *Proceedings of European Conference on the Applications of Evolutionary Computation* (pp. 636–652). Springer, Cham.

Chakraborty, D., Saha, S., & Maity, S. (2015). Training feedforward neural networks using hybrid flower pollination-gravitational search algorithm. In *Proceedings of International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)* (pp. 261–266). IEEE. https://doi.org/10.1109/ABLAZE.2015.7155008

Clerc, M. (2011). Standard PSO 2011 (SPSO). Retrieved from https://www.particleswarm.info/Programs.html

Colorni, A., Dorigo, M., & Maniezzo, V. (1991). Distributed optimization by ant colonies. In *Proceedings of the 1st European Conference on Artificial Life* (Vol. 142, pp. 134–142). https://doi.org/10.1016/S0303-2647(97)01708-5

Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and Exploitation in Evolutionary Algorithms: A Survey. *ACM Computing Surveys*, *45*(3), 35:1-35:33. https://doi.org/10.1145/2480741.2480752

Deneubourg, J. L., & Goss, S. (1989). Collective patterns and decision-making. *Ethology Ecology and Evolution*, *1*(4), 295–311. https://doi.org/10.1080/08927014.1989.9525500

Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., & Chrétien, L. (1991).

The fynamics of collective sorting robot - like ants and ant - like robots. In *Proceedings of the 1st International Conference on Simulation of Adaptive Behavior on From Animals to Animats* (pp. 356–363). MIT Press.

Dorigo, M. (1992). *Optimization, learning and natural algorithms*. Politecnico di Milano.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *26*(1), 29–41. https://doi.org/10.1109/3477.484436

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS 95)* (pp. 39–43). IEEE. https://doi.org/10.1109/MHS.1995.494215

Eiben, A. E., & Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*. https://doi.org/10.1016/j.swevo.2011.02.001

El-Shorbagy, M. A., & Hassanien, A. E. (2018). Particle swarm optimization from theory to applications. *International Journal of Rough Sets and Data Analysis (IJRSDA)*, *5*(2), 1–24.

Emami, H., & Derakhshan, F. (2015). Election algorithm: a new socio-politically inspired strategy. *AI Communications*, *28*(3), 591–603. https://doi.org/10.3233/AIC-140652

Engelbrecht, A. P. (2007). *Computational Intelligence : An Introduction* (2nd ed.). John Wiley & Sons. https://doi.org/10.1002/9780470512517

Fister, I., Yang, X. S., Brest, J., & Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *Elektrotehniski Vestnik/Electrotechnical Review*, *80*(3), 116–122.

Fister, I., Yang, X. S., Fister, D., & Fister, I. (2014). Cuckoo search: A brief literature review. In *Studies in Computational Intelligence* (Vol. 516, pp. 49–62). https://doi.org/10.1007/978-3-319-02141-6_3

Formato, R. A. (2014). Central force optimization algorithm. *Intelligent Systems Reference Library*, *62*, 333–337. https://doi.org/10.1007/978-3-319-03404-1_19

Gandomi, A. H., Yang, X. S., Talatahari, S., & Alavi, A. H. (2013). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, *18*(1), 89–98. https://doi.org/10.1016/j.cnsns.2012.06.009

Gendreau, M., & Potvin, J.-Y. (2018). *Handbook of metaheuristics* (Vol. 57). https://doi.org/10.1007/b101874

Gupta, S., & Deep, K. (2018). A novel random walk grey wolf optimizer. *Swarm and Evolutionary Computation*. https://doi.org/10.1016/j.swevo.2018.01.001

Heidari, A. A., & Pahlavani, P. (2017). An efficient modified grey wolf optimizer with Lévy flight for optimization tasks. *Applied Soft Computing Journal*, *60*, 115–134. https://doi.org/10.1016/j.asoc.2017.06.044

Ho, Y. C., & Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications*, *115*(3), 549–570. https://doi.org/10.1023/A:1021251113462

Holland, J. H. (1975). Adaptation in natural and artificial systems. *Ann Arbor MI University of Michigan Press, Ann Arbor*, 183. https://doi.org/10.1137/1018105

Huan, T. T., Kulkarni, A. J., Kanesan, J., Huang, C. J., & Abraham, A. (2017). Ideology algorithm: a socio-inspired optimization methodology. *Neural Computing and Applications*, *28*, 845–876. https://doi.org/10.1007/s00521-016-2379-4

Hussain, K., Salleh, M. N. M., Cheng, S., & Shi, Y. (2018). Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 1–43.

Husseinzadeh Kashan, A. (2014). League championship algorithm (LCA): An algorithm for global optimization inspired by sport championships. *Applied Soft Computing Journal*, *16*, 171–200. https://doi.org/10.1016/j.asoc.2013.12.005

Jamil, M., Yang, X. S., & Zepernick, H. J. D. (2013). Test functions for global optimization: a comprehensive survey (Book Chapter). In *Swarm Intelligence and Bio-Inspired Computation* (pp. 193–222). Elsevier. https://doi.org/10.1016/B978-0-12-405163-8.00008-9

Kallioras, N. A., Lagaros, N. D., & Avtzis, D. N. (2018). Pity beetle algorithm – a new metaheuristic inspired by the behavior of bark beetles. *Advances in Engineering Software*, *121*, 147–166. https://doi.org/10.1016/j.advengsoft.2018.04.007

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical Report TR06, Erciyes University*, (TR06), 10. https://doi.org/citeulike-article-id:6592152

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, *39*(3), 459–471. https://doi.org/10.1007/s10898-007-9149-x

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57. https://doi.org/10.1007/s10462-012-9328-0

Kashan, A. H. (2009). League championship algorithm: a new algorithm for numerical function optimization. In *Proceedings of International Conference on Soft Computing and Pattern Recognition (SoCPaR 2009)* (pp. 43–48). https://doi.org/10.1109/SoCPaR.2009.21

Kashif, H., Salleh, N., Cheng, S., & Shi, Y. (2018). On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications*, 1–19.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of International Conference on Neural Networks (ICNN'95)* (Vol. 4, pp. 1942–1948 vol.4). IEEE. https://doi.org/10.1109/ICNN.1995.488968

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671–680. https://doi.org/10.1126/science.220.4598.671

Köppen, M., Wolpert, D. H., & Macready, W. G. (2001). Remarks on a recent paper on the "no free lunch" theorems. *IEEE Transactions on Evolutionary Computation*, *5*(3), 295–296. https://doi.org/10.1109/4235.930318

Koziel, S., & Yang, X. S. (2011). *Computational optimization, methods and algorithms*. *Springer*. https://doi.org/10.1007/978-3-642-20859-1

Kulkarni, A. J., Krishnasamy, G., & Abraham, A. (2016). Socio-inspired optimization using cohort intelligence (Book Chapter). In *Cohort Intelligence: A Socio-inspired Optimization Method* (Vol. 114, pp. 9–24). https://doi.org/10.1007/978-3-319-44254-9_2

Kumar, M., Kulkarni, A. J., & Satapathy, S. C. (2017). Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology. *Future Generation Computer Systems*, *81*, 252–272. https://doi.org/10.1016/j.future.2017.10.052

Kuo, H. C., & Lin, C. H. (2013). Cultural evolution algorithm for global optimizations and its applications. *Journal of Applied Research and Technology*, *11*(4), 510–522. https://doi.org/10.1016/S1665-6423(13)71558-X

Li, X., Shao, Z., & Qian, J. (2002). An optimizing method based on autonomous animats: fish-swarm algorithm. *Systems Engineering: Theory and Practice*, *22*(11), 32–38. Retrieved from http://en.cnki.com.cn/Article_en/CJFDTOTAL-XTLL200211006.htm

Luke, S. (2013). *Essentials of metaheuristics*. *Optimization*. https://doi.org/10.1007/s10710-011-9139-0

Lv, W., He, C., Li, D., Cheng, S., Luo, S., & Zhang, X. (2010). Election campaign optimization algorithm. In *Proceedings of International Conference on Computational Science (ICCS 2010)* (Vol. 1, pp. 1377–1386). Elsevier. https://doi.org/10.1016/j.procs.2010.04.153

Lynn, N., & Suganthan, P. N. (2017). Ensemble particle swarm optimizer. *Applied Soft Computing*, *55*, 533–548. https://doi.org/10.1016/j.asoc.2017.02.007

Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software*, *83*, 80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010

Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163–191. https://doi.org/10.1016/j.advengsoft.2017.07.002

Mirjalili, S., & Hashim, S. Z. M. (2010). A new hybrid PSOGSA algorithm for function optimization. In *Proceedings of International Conference on Computer and Information Application (ICCIA 2010)* (pp. 374–377). IEEE. https://doi.org/10.1109/ICCIA.2010.6141614

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Mirjalili, S., Mohd Hashim, S. Z., & Moradian Sardroudi, H. (2012). Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation*, *218*(22), 11125–11137. https://doi.org/10.1016/j.amc.2012.04.069

Moosavian, N., & Kasaee Roodsari, B. (2014). Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. *Swarm and Evolutionary Computation*, *17*, 14–24. https://doi.org/10.1016/j.swevo.2014.02.002

Mortazavi, A., Toğan, V., & Nuhoğlu, A. (2018). Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Engineering*

Applications of Artificial Intelligence. https://doi.org/10.1016/j.engappai.2018.03.003

Mousavirad, S. J., & Ebrahimpour-Komleh, H. (2017). Human mental search: a new population-based metaheuristic optimization algorithm. *Applied Intelligence*, *47*(3), 850–887. https://doi.org/10.1007/s10489-017-0903-6

Niu, B., Zhu, Y., He, X., & Wu, H. (2007). MCPSO: A multi-swarm cooperative particle swarm optimizer. *Applied Mathematics and Computation*, *185*(2), 1050–1062. https://doi.org/10.1016/j.amc.2006.07.026

Odili, J. B., Kahar, M. N. M., & Anwar, S. (2015). African buffalo optimization: a swarm-intelligence technique. In *Proceedings of International Symposium on Robotics and Intelligent Sensors (IRIS2015)* (Vol. 76, pp. 443–448). IEEE.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE*, *22*(3), 52–67. https://doi.org/10.1109/MCS.2002.1004010

Qu, B. Y., Liang, J. J., Wang, Z. Y., Chen, Q., & Suganthan, P. N. (2016). Novel benchmark functions for continuous multimodal optimization with comparative results. *Swarm and Evolutionary Computation*, *26*, 23–34. https://doi.org/10.1016/j.swevo.2015.07.003

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, *43*(3), 303–315. https://doi.org/10.1016/j.cad.2010.12.015

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2012). Teaching-Learning-Based Optimization: an optimization method for continuous non-linear large scale problems. *Information Sciences*, *183*(1), 1–15. https://doi.org/10.1016/j.ins.2011.08.006

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences*, *179*(13), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Ray, T., & Liew, K. M. (2003). Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, *7*(4), 386–396. https://doi.org/10.1109/TEVC.2003.814902

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, *21*(4), 25–34. https://doi.org/10.1145/37402.37406

Reynolds, R. G., & Sverdlik, W. (1994). Problem solving using cultural algorithms. In *Proceedings of the 1st IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence* (pp. 645–650). IEEE. https://doi.org/10.1109/ICEC.1994.349983

Saka, M. P., Doğan, E., & Aydogdu, I. (2013). Analysis of swarm intelligence-based algorithms for constrained optimization (Book Chapter). In *Swarm Intelligence and Bio-Inspired Computation* (pp. 25–48). https://doi.org/10.1016/B978-0-12-405163-8.00002-8

Salgotra, R., & Singh, U. (2017). Application of mutation operators to flower pollination algorithm. *Expert Systems with Applications*, *79*, 112–129. https://doi.org/10.1016/j.eswa.2017.02.035

Satapathy, S., & Naik, A. (2016). Social group optimization (SGO): a new population evolutionary optimization technique. *Complex & Intelligent Systems*, *2*(3), 173–203. https://doi.org/10.1007/s40747-016-0022-8

Shi, J., & Zhang, Q. (2018). A new cooperative framework for parallel trajectory-based metaheuristics. *Applied Soft Computing*, *65*, 374–386. https://doi.org/10.1016/j.asoc.2018.01.022

Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC 99)* (Vol. 3, p. 1945–1950 Vol. 3). IEEE. https://doi.org/10.1109/CEC.1999.785511

Siddique, N., & Adeli, H. (2015). Nature inspired computing: an overview and some future directions. *Cognitive Computation*, *7*(6), 706–714. https://doi.org/10.1007/s12559-015-9370-8

Silberholz, J., & Golden, B. (2010). Comparison of metaheuristics. *Handbook of Metaheuristics*, 625–640.

Slowik, A., & Kwasnicka, H. (2018). Nature inspired methods and their industry applications – swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics*, *14*(3), 1004–1015. https://doi.org/10.1109/TII.2017.2786782

Socha, K., & Dorigo, M. (2008). Ant colony optimization for continuous domains. *European Journal of Operational Research*, *185*(3), 1155–1173. https://doi.org/10.1016/j.ejor.2006.06.046

Sörensen, K., Sevaux, M., & Glover, F. (2018). A history of metaheuristics. *Handbook of Heuristics*, 1–18.

Talbi, E. G. (2009). *Metaheuristics: from design to implementation*. Wiley. https://doi.org/10.1002/9780470496916

Ursem, K. R. (2003). *Models for Evolutionary Algorithms and their Applications in System Identification and Control Optimization*. University of Aarhus, Denmark.

Wang, H., Wang, W., Zhou, X., Sun, H., Zhao, J., Yu, X., & Cui, Z. (2017). Firefly algorithm with neighborhood attraction. *Information Sciences*, *382–383*, 374–387. https://doi.org/10.1016/j.ins.2016.12.024

Wang, Y. (2010). A sociopsychological perspective on collective intelligence in metaheuristic computing. *International Journal of Applied Metaheuristic Computing*, *1*(1), 110–128. https://doi.org/10.4018/jamc.2010102606

Weise, T. (2009). Global optimization algorithms–theory and application. *Self-Published*, *1*, 820. https://doi.org/doi=10.1.1.64.8184

Weise, T., Zapf, M., Chiong, R., & Nebro, A. J. (2009). Why is optimization difficult? In R. Chiong (Ed.), *Nature-Inspired Algorithms for Optimisation* (pp. 1–50). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-00267-0_1

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82. https://doi.org/10.1109/4235.585893

Xu, Y., Cui, Z., & Zeng, J. (2010). Social emotional optimization algorithm for nonlinear constrained optimization problems. *Swarm, Evolutionary, and Memetic Computing. Springer Berlin Heidelberg*, 583–590.

Yadav, A., Deep, K., Kim, J. H., & Nagar, A. K. (2016). Gravitational swarm optimizer for global optimization. *Swarm and Evolutionary Computation*, *31*, 64–89. https://doi.org/10.1016/j.swevo.2016.07.003

Yang, X. S. (2008). *Firefly algorithm for multimodal optimization*. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Yang, X. S. (2009). Firefly algorithms for multimodal optimization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5792 LNCS*, 169–178. https://doi.org/10.1007/978-3-642-04944-6_14

Yang, X. S. (2010a). A new metaheuristic bat-inspired algorithm (Book Chapter). In *Studies in Computational Intelligence* (Vol. 284, pp. 65–74). https://doi.org/10.1007/978-3-642-12538-6_6

Yang, X. S. (2010b). *Engineering optimization: an introduction with metaheuristic applications*. Wiley. https://doi.org/10.1002/9780470640425

Yang, X. S. (2010c). *Nature-inspired metaheuristic algorithms* (2nd ed.). Luniver press. https://doi.org/10.1016/B978-0-12-416743-8.00005-1

Yang, X. S. (2011). Metaheuristic optimization: algorithm analysis and open problems. In *Proceedings of International Symposium on Experimental Algorithms* (pp. 21–32).

Yang, X. S. (2012). Flower pollination algorithm for global optimization. In *Proceedings of International Conference on Unconventional Computing and Natural Computation.* (pp. 240–249). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-32894-7_27

Yang, X. S. (2013). Metaheuristic optimization: nature-inspired algorithms and applications. *Studies in Computational Intelligence*. https://doi.org/10.1007/978-3-642-29694-9-16

Yang, X. S. (2014). *Nature-inspired optimization algorithms*. Elsevier. https://doi.org/10.1016/C2013-0-01368-0

Yang, X. S., & Deb, S. (2009). Cuckoo search via lévy flights. In *Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC)* (pp. 210–214). https://doi.org/10.1109/NABIC.2009.5393690

Yang, X. S., & Deb, S. (2014). Cuckoo search: recent advances and applications. *Neural Computing and Applications*, *24*(1), 169–174. https://doi.org/10.1007/s00521-013-1367-1

Yang, X. S., Deb, S., Fong, S., He, X., & Zhao, Y. X. (2016). From swarm intelligence to metaheuristics: nature-inspired optimization algorithms. *Computer*, *49*(9), 52–59. https://doi.org/10.1109/MC.2016.292

Yang, X. S., Deb, S., & He, X. (2013). Eagle strategy with flower algorithm. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI 2013)* (pp. 1213–1217). https://doi.org/10.1109/ICACCI.2013.6637350

Yang, X. S., Deb, S., Loomes, M., & Karamanoglu, M. (2013). A framework for self-tuning optimization algorithm. *Neural Computing and Applications*, *23*(7–8), 2051–2057. https://doi.org/10.1007/s00521-013-1498-4

Yang, X. S., Deb, S., Zhao, Y.-X., Fong, S., & He, X. (2017). Swarm intelligence: past, present and future. *Soft Computing*, *22*(18), 5923–5933. https://doi.org/10.1007/s00500-017-28105

Yang, X. she. (2010). Firefly algorithm, Levy flights and global optimization (Book Chapter). In *Research and Development in Intelligent Systems* (pp. 209–218). https://doi.org/10.1007/978-1-84882-983-1

Yaseen, Z. M., Ebtehaj, I., Bonakdari, H., Deo, R. C., Danandeh Mehr, A., Mohtar, W. H. M. W., Diop, L., El-shafie, A., & Singh, V. P. (2017). Novel approach for streamflow forecasting using a hybrid ANFIS-FFA model. *Journal of Hydrology*, *554*, 263–276. https://doi.org/10.1016/j.jhydrol.2017.09.007

Zhang, L., Shan, L., & Wang, J. (2017). Optimal feature selection using distance-based discrete firefly algorithm with mutual information criterion. *Neural Computing and Applications*, *28*(9), 2795–2808. https://doi.org/10.1007/s00521-016-2204-0

Zhang, Y., & Wu, L. (2011). A hybrid TS-PSO optimization algorithm. *Journal of Convergence Information Technology*, *6*(5), 169–174. https://doi.org/10.4156/jcit.vol6.issue5.18

Zong, W. G., Joong, H. K., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. *Simulation*, *76*(2), 60–68. https://doi.org/10.1177/003754970107600201

APPENDIX A
METAHEURISTICS USED FOR COMPARISON

In this appendix, full information about the metaheuristics used in chapter four is displayed. This information consists of (Metaheuristic Title, Name of Author(s), the year of publication, the link of the website, the code download link, and the main pseudocode). As mentioned in chapter four, there are five metaheuristics which have been used for comparison; they are (Particle Swarm Optimization "PSO", Artificial Bees Colony "ABC", Flower Pollination Algorithm "FPA", Grey Wolf Optimizer "GWO", and Firefly Algorithm "FA"). The results obtained from these algorithms by executing their original codes, which have been published online on their respective websites.

Title:	Particle Swarm Optimization (PSO)
Author(s) :	Eberhart and Kennedy
Year:	1995
Inspiration Source	based on the social behaviour of birds (flocking) and fishes
Website Link	https://www.particleswarm.info
Download Link	https://www.particleswarm.info/SPSO2011_matlab.zip

Pseudocode

Algorithm 1: SPSO

```
1.  Input: $\#Particles, \#MaxItr, c_1, c_2$
2.  Initialize all $\boldsymbol{Particles}$
3.  Calculate inertia weight via $eq. 5$
4.  Evaluate the fitness for all $\boldsymbol{Particles}$
5.  While $(itr \leq \#MaxItr)$
6.      For each particle $p$ in $\boldsymbol{Particles}$
7.          Calculate the velocity for $p$ via $eq. 3$
8.          Calculate the new position via $eq. 2$
9.          Evaluate the fitness for all $\boldsymbol{Particles}$
10.         If $\boldsymbol{p.Fitness}$ is better than $\boldsymbol{pBest.Fitness}$ Then $pBest$ = p
11.     Next p
12.     $gBest$ = Determine the best in $\boldsymbol{Particles}$
13. End While
```

Title:	Artificial Bees Colony (ABC)
Author(s) :	Dervis KARABOGA
Year:	2005
Inspiration Source	It is inspired by the behavior of honey bees when seeking a quality food source.
Website Link	https://abc.erciyes.edu.tr
Download Link	https://abc.erciyes.edu.tr/form.aspx

Pseudocode

Algorithm 2: ABC

```
1.  Input   : N (Scouts), N (Experienced), N (Onlooker), N (Bees), ItrMax
2.  Output  : Optimal Bee Position
3.  Generate Initial Population of N random positions;
4.  While Stop Condition not met Do
5.      Evaluate Individual bee position given the fitness function
6.      Select the best position; Elitist, according to fitness values
7.      Divide the swarm according to fitness of best into Experienced, Onlooker, Scout
8.      For each Experienced Do
9.          Update the positions of experienced bees and determine the current global best.
10.     For each Onlooker Do
11.         Select one of the experienced bees as an elite one using RouletteWheel approach
12.         Update the position of onlooker bees
13.     For each Scout Do
14.         Walk randomly around the search space
15.         Update the position of scout bees
16. End while
```

Title:	Flower Pollination Algorithm (FPA)
Author(s) :	Xin-She Yang
Year:	2012
Inspiration Source	Flower pollination process is associated with the transfer of pollen by using pollinators such as insects, birds, bats, … .
Website Link	https://www.mathworks.com/matlabcentral/fileexchange/45112-flower-pollination-algorithm
Download Link	https://www.mathworks.com/login?uri=https%3A%2F%2Fwww.mathworks.com%2Fmatlabcentral%2Ffileexchange%2F45112-flower-pollination-algorithm&form_type=community
Pseudocode	

Algorithm 3: FPA

```
1.  Initialize a population of n flowers/pollen gametes with random solutions
2.  Define Objective Function $f(x)$, $x_i = \{x_1, x_2, x_3, \dots, x_d\}$
3.  Find the best solution $g_*$ in the initial population
4.  Define a switch probability $p \in [0,1]$
5.  While ($t \leq MaxGen$)
6.      For each flower in the population
7.          If $rand < p$
8.              Draw a $d-$ dimensional step vector $L$ which obeys a Levy distribution
9.              Global pollination via $x_i^{t+1} = x_i^t + L(g_* - x_i^t)$
10.         Else
11.             Draw $\epsilon$ from a uniform distribution in [0,1]
12.             Randomly choose $j$ and $k$ among all the solutions
13.             Do local pollination via $x_i^{t+1} = x_i^{t+1} + \epsilon(x_i^t - x_k^t)$
14.         End if
15.         Evaluate new solutions
16.         If new solutions are better, update them in the population
17.     End for
18.     Find the current best solution $g_*$
19. End While
```

Title:	Gray Wolf Optimizer (GWO)
Author(s) :	Ali Mirjalili
Year:	2014
Inspiration Source	It simulates the leadership hierarchy and hunting mechanism of gray wolves in nature.
Website Link	http://www.alimirjalili.com/GWO.html
Download Link	http://www.alimirjalili.com/SourceCodes/GWO.zip

Pseudocode

Algorithm 4: GWO

```
1.  Initialize a population of n wolves
2.  Define Objective Function $f(x)$, $x_i = \{x_1, x_2, x_3, \ldots, x_d\}$
3.  Initialize $a, A$, and $C$
4.  Calculate the fitness of each search agent
5.  $X_a = the\ best\ search\ agent$
6.  $X_b = the\ second\ best\ search$
7.  $X_\delta = the\ third\ best\ search\ agent$
8.  While ($t \leq MaxGenerations$)
9.          For each search agent
10.             Update the position of current search agent
11.         End For
12.         Update $a, A$, and $C$
13.         Calculate the fitness of all search agents
14.         Update $X_a, X_b, X_\delta$
15.         $t = t + 1$
16. End While
17. Return $X_a$
```

Title:	Firefly Algorithm (FFA)
Author(s) :	Xin-She Yang
Year:	2009
Inspiration Source	It is based on the behavior and light flashing patterns of tropical fireflies
Website Link	https://www.mathworks.com/matlabcentral/fileexchange/29693-firefly-algorithm
Download Link	https://www.mathworks.com/login?uri=https%3A%2F%2Fwww.mathworks.com%2Fmatlabcentral%2Ffileexchange%2F29693-firefly-algorithm&form_type=community

Pseudocode

Algorithm 4: FFA

```
1.  Initialize the swarm of n Fireflies
2.  Define Objective Function $f(x)$, $x_i = \{x_1, x_2, x_3, \dots, x_d\}$
3.  Initialize $\alpha, \beta_0$ and $\delta$
4.  Calculate the fitness and the light intensity of each firefly and
5.  While ($t \leq MaxGenerations$)
6.      For each firefly ($f_i$) in the swarm
7.          For each firefly ($f_j$) in the swarm
8.              If the intensity of ($f_i$) is less than the intensity of ($f_j$)
9.                  Calculate the distance between $f_i$ and $f_j$
10.                 Calculate the attractiveness between $f_i$ and $f_j$
11.                 Update the position of ($f_i$)
12.                 Evaluate the new solution and update the light intensity
13.             End if
14.         End For
15.     End For
16.     Rank the swarm and find the current best firefly
17.     $t = t + 1$
18. End While
19. Return best firefly
```

APPENDIX B
MULTI-SWARM PARTICLE SWARM OPTIMIZATION (MPSO)

Several metaheuristics have been previously proposed and several improvements have been implemented as well. Most of these methods were either inspired by nature or by the behaviour of certain swarms such as birds, ants, bees, or even bats. In the metaheuristics, two key components (exploration and exploitation) are significant and their interaction can significantly affect the efficiency of a metaheuristic. However, there is no rule on how to balance these important components. In this thesis, a new balancing mechanism based on multi-swarm approach is proposed for balancing exploration and exploitation in metaheuristics. The aim of this appendix is to evaluate the meeting room approach on well-known metaheuristic, which is Particle Swarm Optimization (PSO) algorithm.

C-1 Challenges of PSO

The PSO is a simple social model which has attracted several research interests since inception in 1995 due to its ease of implementation and simplicity (Eberhart and Kennedy, 1995). Although PSO has undergone several developmental modifications, it is still prone to the following problems which demand to be addressed in future studies:

- Premature convergence: The PSO ends up searching early best solutions and this is predominant in multimodal functions.

- Convergence speed: The PSO establishes the best solution in the early search stage but get stagnated in the process of exploiting the global solution.

- Quality of solution. The PSO produces low-quality solutions due to inherent problem complexity, multimodality, and discontinuity. Uncertainty of solutions. The stochastic nature of PSO makes it produce different solutions in different runs.

- Update strategy: The PSO has a simple solution update strategy and hence, cannot achieve better solutions in complex environments.

C-2 PSO based Meeting Room Approach

The core idea of multi-swarm is the interaction between several groups while searching for a solution. Many multi-swarm schemes have been proposed, each idea inspired by a natural behaviour. In this chapter, a new cooperative multi-swarm scheme inspired by the human social behaviour is proposed. Here, the interaction is between groups of people known as 'Clans' and their leaders. The proposed scheme consists of several swarms called 'clans'; each clan consists of several solutions which represent the group members. The best member is designated as the leader of a clan; the leader controls the members of the clan in terms of the time to move or where to explore. Figure C-1 depicts the structure of the individual swarm.

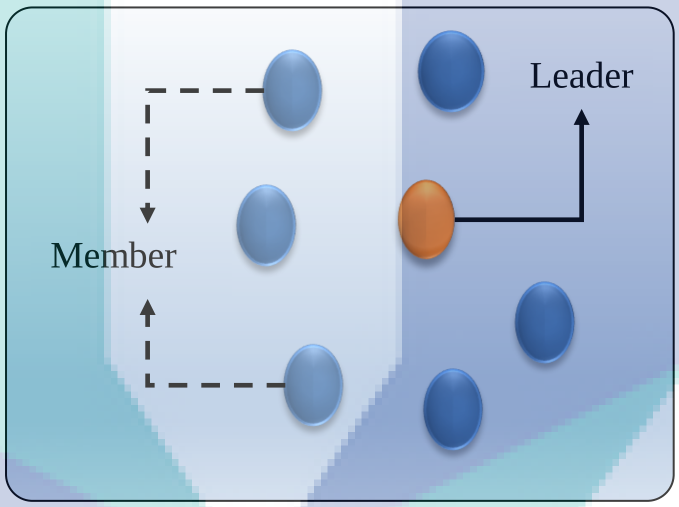


Figure B. 1 The structure of the individual swarm

In each generation, the leaders meet in a room where the overall best leader update the positions of the other normal leaders based on his own positional information. This behaviour of knowledge sharing helps to balance the exploration stage with the exploitation stage of the PSO. The new multi-swarm approach is called 'Meeting Room Approach' (MRA). Figure C-2 depicts the PSO based MRA model. In this figure, each clan – or swarm – performs a single PSO search, including positional and velocity updating, as well as new local population generation. Having established the new generations for all the clans, each clan sends the leader 'best solution' to the meeting room. The best among all the leaders in the meeting room is selected as the overall best leader. The new overall best leader shares his positional information with the ordinary leaders using the following equations:

$$w^{Ln} = \left(\frac{w^{Lg} - w^{Ln}}{Itr}\right) \times rand() \qquad \text{B.1}$$

$$v_i^{Ln}(t+1) = w^{Ln} \times v_i^{Ln}(t) + rc\left(P_g{}^{L} - P_n{}^{L}(t)\right) \qquad \text{B.2}$$

$$x_i^{Ln}(t+1) = x_i^{Ln}(t) + v_i^{Ln}(t) \qquad \text{B.3}$$

Where Ln represents the normal leaders, Lg represents the overall best leader, $x_i{}^{L}$ represents the position of the normal leader, $v_i{}^{Ln}$ represents the velocity of the normal leader, w^{Lg} and w^{Ln} represent the inertia weight of the best leader and the normal leader, respectively.

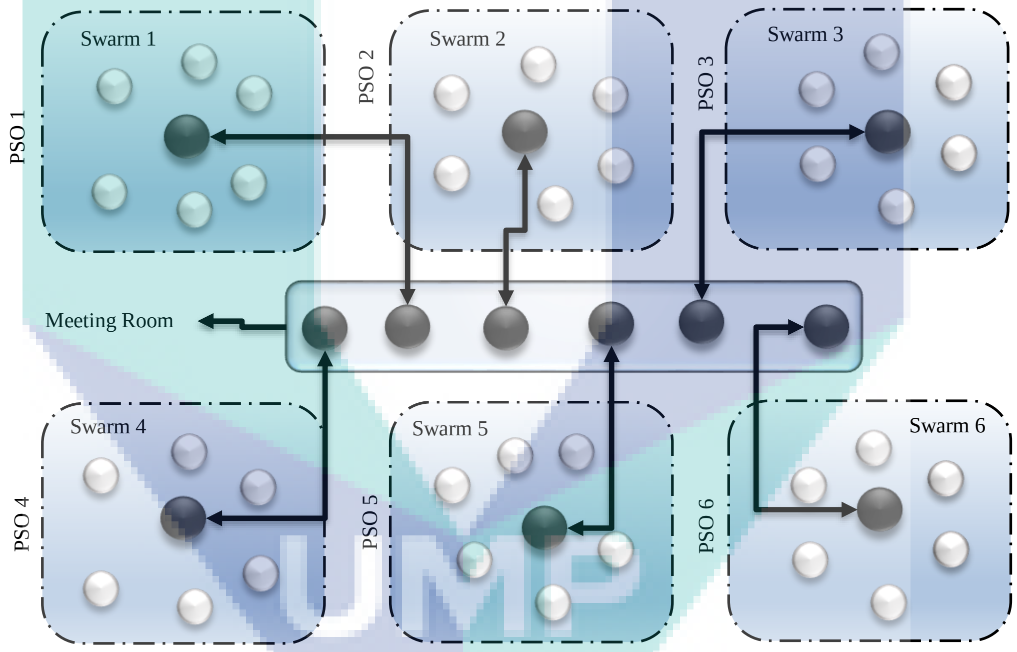


Figure B. 2 PSO based MRA

After each generation, a new leader is chosen for each swarm due to the changes in the positions of the members. The new equation of the inertia in the meeting room controls the exploration of the search algorithm. The pseudo-code of the proposed Multi-Swarm Particle Swarm Optimisation (MPSO) algorithm is presented in Figure C-3.

It is worth to mention that the new version of PSO algorithm (MPSO) does not utilise the same equations of MRA which have been discussed in chapter 3. MPSO uses the core idea of MRA and applied it based on the behaviour of PSO. In other words, there is no mention to ΔPos or Ψ variable, they are related to NPO algorithm. Therefore, any metaheuristic uses MRA in future, should develop new specific equations for the interactions and the information sharing in the meeting room.

Algorithm: MPSO

```
1.   Input: $\#Swarm, \#Particles, \#MaxItr, c_1, c_2, \#Dim$
2.   Initialize all $\boldsymbol{Leaders}$ and their $\boldsymbol{Swarms}$
3.   Calculate inertia weight via $eq. 5$
4.   Evaluate the fitness for all $\boldsymbol{Particles}$
5.   While $(itr \leq \#MaxItr)$
6.     For each c in $\boldsymbol{Clans}$
7.         For each particle $p$ in $\boldsymbol{Particles}$
8.             Calculate the velocity for $p$
9.             Calculate the new position
10.            Evaluate the fitness for all $\boldsymbol{Particles}$
11.            If $p.Fitness$ is better than $pBest.Fitness$ Then
12.                $pBest$ = p
13.            End
14.        Next $p$
15.    Next $c$
16.    Determine the best $\boldsymbol{Leader}$ among all $\boldsymbol{Leaders}$
17.    Update the inertia weight of the Clan c via $eq\ B.1$
18.    Update the velocity of the $Leader_c$ via $eq\ B.2$
19.    Update the position of the $Leader_c$ via $eq\ B.3$
20.    Determine the $\boldsymbol{gLeader}$ ever as the global best.
21.  End While
22.  Return $\boldsymbol{Best\ Leader}$
```

Figure B. 3 Pseudo-code of MPSO

B-3 Results and Discussion

This results of the benchmarking evaluations commonly used in the evolutionary literature are presented in this section (Jamil et al., 2013). Each test function varies in terms of modality (unimodal and multimodal) and the number of dimensions (fixed and dynamic). There are four test functions used in this section for evaluating the MPSO. These functions are (Sphere, Grewank, Rastirign, and Ackley).

The performance of the MPSO was evaluated by comparing with that of original PSO (SPSO)(Y. Shi and Eberhart, 1999) and Master-Slave PSO (MCPSO)(Niu et al., 2007). The parameters used for SPSO were recommended by (Y. Shi and Eberhart, 1999) with asymmetric initialization method and a linearly decreasing w (changes from 0.9 to

0.4). Several swarms of SPSO were involved in the MPSO and MCPSO (as clans and slaves respectively) during the benchmark function optimization. Both MPSO and MCPSO have the same parameter settings as SPSO1. To investigate the efficiency of the proposed MPSO, different population sizes with different dimensions were used for each function. The maximum iteration number was set to 500, corresponding to the dimensions 100, 500, and 1000. The experiments were conducted for a total of 30 settings. Table C-1 presents the parameters setting for all the evaluated algorithms.

Table B. 1 Parameters Settings

Algorithm	Parameter	Value
SPSO	W No. of Swarms c_1,c_2 Swarm Size	0.9 – 0.4 1 1.5 50
MCPSO	W No. of Slaves c_1,c_2, c_3 Swarm Size	0.9-0.6 5 1.5 50
MPSO	w^{Ln} w^{Lg} c_1,c_2 No. of Clans Clan Size	0.8 – 0.5 0.9 – 0.7 1.5 5 10

The best and mean fitness values of the particles after 30 experimental runs over 4 benchmark functions are presented in Table B-2. From the table, MPSO performed better than the benchmarking algorithms in almost all the cases. Generally analysing the table, MPSO has 5 swarms; each swarm consists of 10 particles but only 5 particles are interacting in the meeting room. Hence, it can be said that the MPSO has less computational complexity and a better performance in terms of finding the best solution. Figure B-4 and Figure B-5 illustrate the ability of the MPSO to evolve in situations that the algorithms may have been trapped.

Table B. 2 Results for benchmark test functions

Dim	f_n	Algorithm	Best	Mean	S.D
100	f_1	SPSO	2.5457521	2.7647845	0.0784516
		MCPSO	0.9854126	1.0154784	0.0014784
		MPSO	**0.0007845**	**0.0008748**	**0.0000184**
	f_2	SPSO	0.0884741	0.0964587	0.0078478
		MCPSO	0.0078414	0.0087789	0.0009874
		MPSO	**0.0000897**	**0.0000997**	**0.0000658**
	f_3	SPSO	21.695847	27.947512	0.0847896
		MCPSO	2.0018977	2.6647845	0.0078487
		MPSO	**0.0004687**	**0.0045214**	**0.0000144**
	f_4	SPSO	16.4875218	26.110161	0.0238484
		MCPSO	1.99847	2.5869124	0.0084578
		MPSO	**0.0002648**	**0.0017636**	**0.0000584**
500	f_1	SPSO	7.2456571	2.7647845	0.0784516
		MCPSO	2.4859157	1.0154784	0.0014784
		MPSO	**0.0026472**	**0.0008748**	**0.0000184**
	f_2	SPSO	1.2785781	0.0964587	0.0078478
		MCPSO	0.9045472	0.0087789	0.0009874
		MPSO	**0.0041816**	**0.0000997**	**0.0000658**
	f_3	SPSO	48.995751	27.947512	0.0847896
		MCPSO	7.2214945	2.6647845	0.0078487
		MPSO	**0.0784457**	**0.0045214**	**0.0000144**
	f_4	SPSO	37.125475	26.110161	0.0238484
		MCPSO	3.35847	2.5869124	0.0084578
		MPSO	**0.1778499**	**0.0017636**	**0.0000584**
1000	f_1	SPSO	16.422422	2.7647845	0.0784516
		MCPSO	4.7923729	1.0154784	0.0014784
		MPSO	**1.0749752**	**0.0008748**	**0.0000184**
	f_2	SPSO	3.0899761	0.0964587	0.0078478
		MCPSO	5.2574914	0.0087789	0.0009874
		MPSO	**0.9177297**	**0.0000997**	**0.0000658**
	f_3	SPSO	21.695847	27.947512	0.0847896
		MCPSO	2.0018977	2.6647845	0.0078487
		MPSO	**0.9563589**	**0.0045214**	**0.0000144**
	f_4	SPSO	16.4875218	26.110161	0.0238484
		MCPSO	1.99847854	2.5869124	0.0084578
		MPSO	**0.48758311**	**0.0017636**	**0.0000584**

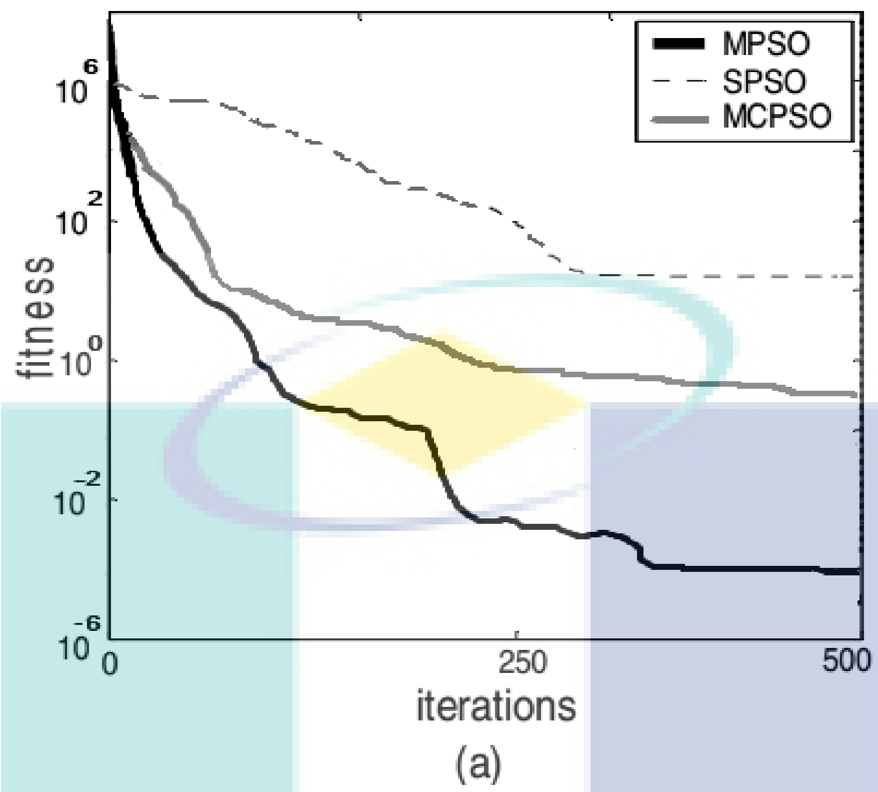


Figure B. 4 Convergence curve for Sphere function

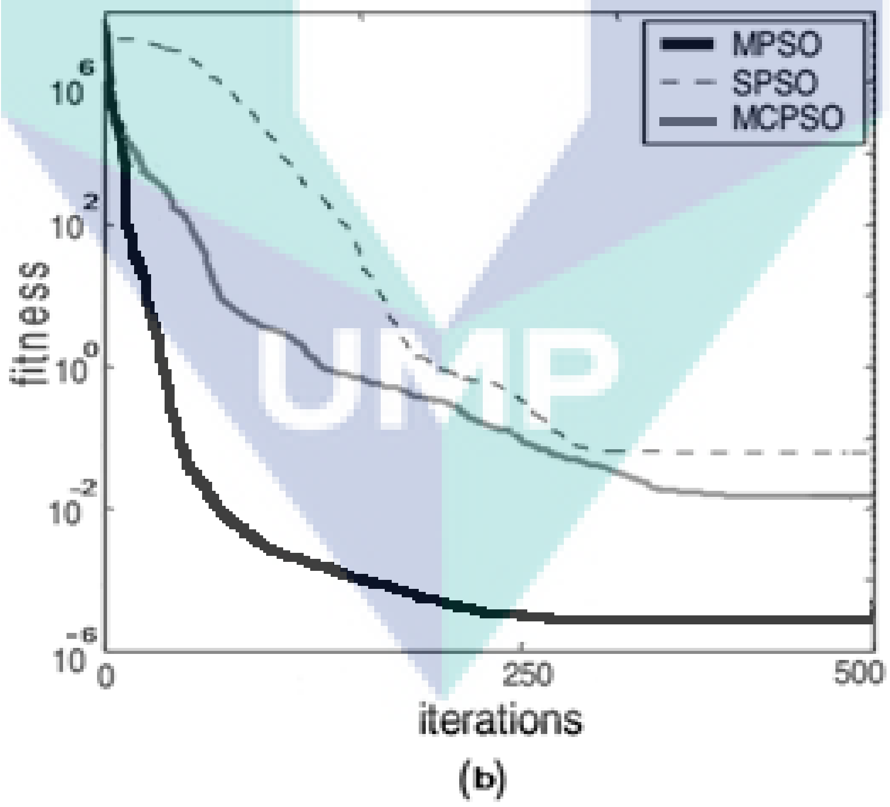


Figure B. 5 Convergence curve for Griewank function