People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

MOHAMED KHIDER UNIVERSITY, BISKRA

FACULTY OF EXACT SCIENCES and SCIENCES of NATURE and LIFE

DEPARTMENT OF MATHEMATICS

Thesis Submitted in Partial Execution of the Requirements of the Degree of

DOCTOR OF SCIENCES

In the field of: *Applied Mathematics*

Option: *Numerical Analysis and Optimization*

Presented by:

OUAAR Fatima

Titled:

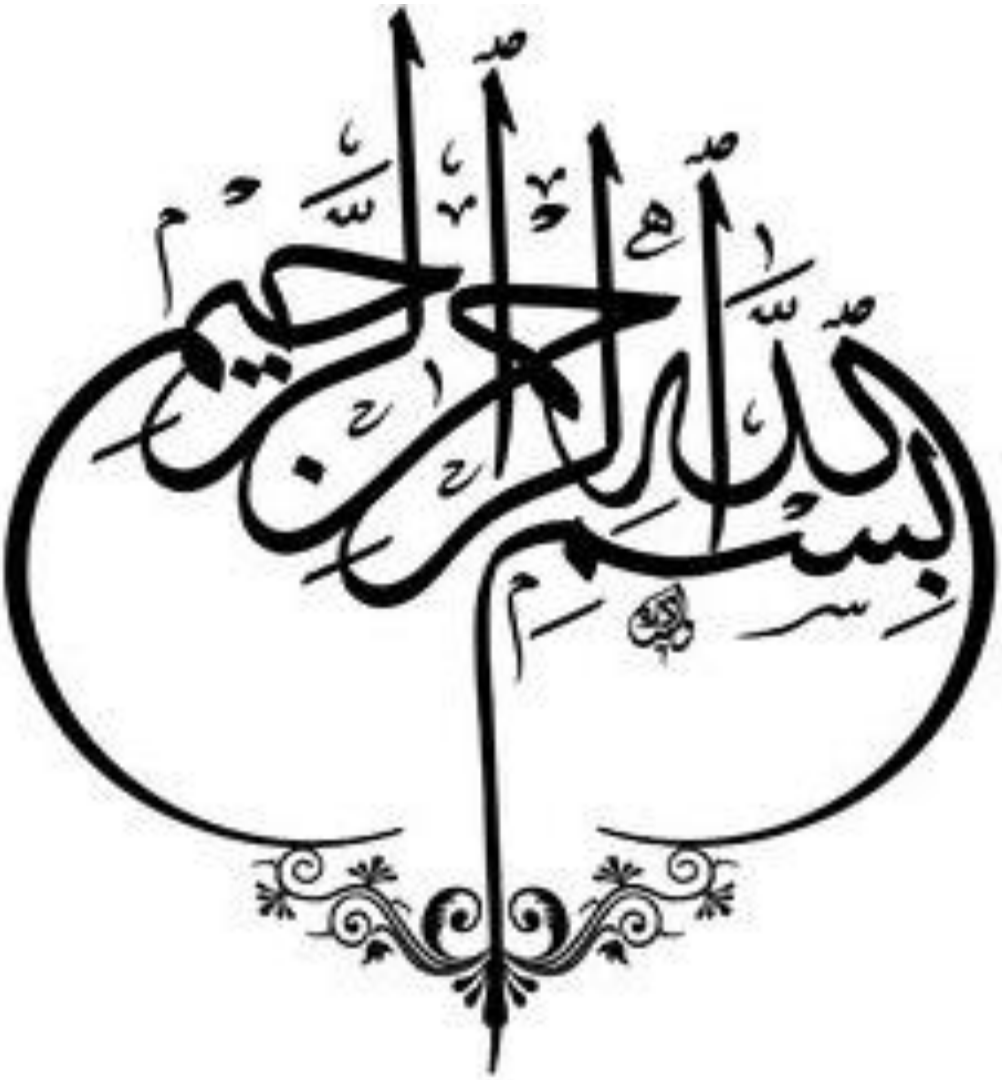
Application of Metaheuristics in Solving Initial Value Problems (IVPs).

A thesis Directed by **KHELIL Naceur** and Co-directed by **BOUDJEMAA Redouane.**

Examination Committee Members

Pr.	MOKHTARI	Zohir	Profs	Biskra University	Chairman.
Pr.	KHELIL	Naceur	Profs	Biskra University	Supervisor.
Dr.	BOUDJEMAA	Redouane	MCA	Blida1 University	Co-Supervisor.
Dr.	REZZOUG	Imad	MCA	Oumelbouaghi University	Examiner.
Dr.	MERAD	Ahcene	MCA	Oumelbouaghi University	Examiner.

2020/2021.


بسم الله الرحمن الرحيم

و في الصبر مشقة يعقبها فرح جميل...

فالحمد لله حمدا دائما ابدا ...

بسم الله الرحمن الرحيم:

"فاصبر ان وعد الله حق و لا يستخفنك الذين لا يوقنون"

سورة الروم - الاية 60-

" الحمد لله الذي هدانا لهذا و ما كنا لنهتدي لولا ان هدانا الله"

سورة الاعراف - الاية 45-

Dedication

To my **P**arents,

To my **H**usband, **T**o my **C**hildren,

To my **B**rothers, **S**isters and their **K**ids,

To all my **F**amily,

To all **T**hose who are **D**ear to **M**e.

Acknowledgements

The realization of this thesis is the culmination of a long academic career, often laborious and during which, despite the pitfalls, I have always been driven by the desire to finish, or rather to finish.

First, I thank **God** the Almighty for helping me to succeed in my quest and giving me courage and patience to realize this work.

The accomplishment of this work could not have been achieved without the support and collaboration of many people whom I wish to sincerely thank, words cannot thank them enough, I would just extend profound thanks to:

First of all, I would like to express my deepest gratitude to my supervisors **Profs. KHELIL Naceur** my thesis director and whose enthusiasm and generosity gave me a taste for research. He is thanked for his patience, his unfailing availability to listen to my questions and his help in finding the answers. For all these reasons, I thank him warmly.

Also I am deeply grateful to my Co-supervisor **Dr. BOUDJEMAA Redouane**, for his valuable support and scholarly advices throughout the entirety of my PhD studies. I appreciate sincerely all the contributions of his time and inspirational expertise, and for instilling in me a greater understanding of applied mathematics. Without his constructive comments, patience and knowledge, this thesis would have not been possible. Thank you for integrating me in the metaheuristics subject.

My thanks goes to the members of the jury for being patient and for having judged my work which I hope to be fruitful. To begin with **Profs. MOKHTARI Zohir**, president of the jury and **Dr. REZZOUG Imad** and **Dr. MERAD Ahcen**, examiners, I express to them my deep gratitude. Really thank you for your interest in my work and for accepting to review my thesis, for your insightful comments and encouragement, and for the hard questions which incited me to enhance my research from different perspectives.

I would also express great gratitude toward my parents, whom supported me throughout my university studies. They know how much they matter to me: **Mom, Dad**. They gave me confidence when I needed it most, they allowed me to continue this work without ever

giving up. For all this and so much more, I will never thank them enough. Your support has been very valuable to me.

I would also like to thank my dear **Brothers** and **Sisters**, with whom I was able to give myself a few discussions and adventures; to which I dedicate all these hours of work devoted to my thesis at the expense of the time I should have spent in their company. They remain my greatest source of motivation and those to whom I owe, in the first place, the realization of this thesis.

I would also like to pass a special thanks to my little family: **My husband** and **My children** for their encouragement and patience over the past few years. As for all the emotional support, understanding, caring and never-ending confidence. I would definitely not be where I am today without their efforts and I thank them deeply.

I would also like to say a word to all those with whom I have had the pleasure of collaborating: **My friends** and **My comrades**. This word, whatever the moments of tension, and for all the moments of jubilation, joy or all the digressions:

Thank You

I have definitely forgotten other people. But I am sure they will forgive me and I am sure they will share with me this moment of euphoria so much expected.

Fatima

Summary

Title : Application of Metaheuristics in Solving Initial Value Problems (IVPs) .

Some differential equations admit analytic solutions given by explicit formulas. However, in most other case only approximated solutions can be found. Several methods are available in the literature to find approximate solutions to differential equations. Numerical methods form an important part of solving IVP in ODE, most especially in cases where there is no closed form of solutions.

The present dissertation focus the attention toward solving IVP by transforming it to an optimization approach which can be solved through the application of non-standard methods called Metaheuristic. By transforming the IVP into an optimization problem, an objective function, which comprises both the IVP and initial conditions, is constructed and its optimum solutions represents an approximative solution of the IVP.

The main contribution of the present thesis is divided in twofold. In the one hand, we consider IVPs as an optimization problem when the search of the optimum solution is performed by means of MAs including ABC, BA and FPA and a set of numerical methods including Euler methods, Runge–Kutta methods and predictor–corrector methods. On the other hand, we propose a new MA called Fractional Lévy Flight Bat Algorithm (**FLFBA**) (which is an improvement of the BA, based on velocity update through fractional calculus and local search procedure based on a Lévy distribution random walk). We illustrates its computational efficiency by comparing its performance with the previous methodds in solving the bacterial population growth models (both the logistic growth model and the exponential growth model).

Key − Words :Initial Value Problem (IVP), Optimization problem, Exponential problem, Logistic problem, FLFBA, Numerical methods, Metaheuristic algorithms.

ملخص

العنوان: تطبيق الخوارزميات Metaheuristic في حل مسائل القيمة الأولية.

تسمح بعض المعادلات التفاضلية بإيجاد حلول تحليلية من خلال الصيغ الصريحة، إلا أنه في الكثير من الحالات يتم إيجاد حلول تقريبية فقط، وفي هذا الإطار تتوفر الأدبيات على عدة طرق لإيجاد حلول تقريبية للمعادلات التفاضلية، إذ تشكل الطرق العددية جزءا مهما من حل مسائل القيمة الأولية في المعادلات التفاضلية العادية وخاصة في الحالات التي لا يوجد فيها صيغة صريحة للحلول.

تركز الأطروحة الحالية على حل مسائل القيمة الأولية من خلال تحويلها إلى منهج التعظيم حيث يمكن حله من خلال تطبيق طرق غير عادية تسمى Metaheuristic. بمعنى أنه يتم تحويل مسائل القيمة الأولية إلى مسالة تعظيم، بإنشاء دالة موضوعية، والتي تشمل كل من مسالة القيمة الأولية و الشروط الابتدائية لها ، حيث تمثل حلولها العظمى حلاً تقريبيًا لمسائل القيمة الأولية.

تنقسم الإشكالية الرئيسية للأطروحة إلى جزئين: يركز الجزء الأول على اعتبار مسائل القيمة الأولية كمشكلة تعظيم حيث يتم إجراء البحث عن الحل الأعظم عن طريق خوارزميات Metaheuristic بما في ذلك خوارزمية مستعمرة النحل الاصطناعية وخوارزمية الخفاش وخوارزمية تلقيح الأزهار و مجموعة من الطرق العددية المعروفة بما في ذلك طريقة أويلر وطريقة رونج – كوتا وطريقة التنبؤ – المصحح. و يستعرض الجزء الثاني من الاشكالية اقتراح خوارزمية جديدة تسمى (Fractional Lévy Flight Bat Algorithm) والتي تعد تحسينًا لخوارزمية الخفاش ، استنادًا إلى تحديث السرعة من خلال حساب التفاضل والتكامل الكسري وإجراء البحث المحلي على أساس مسار عشوائي لتوزيع ليفي. نوضح فعاليتها الحسابية من خلال مقارنة أدائها مع الطرق السابقة في حل نماذج النمو السكاني البكتيري في كل من نموذج النمو اللوجستي ونموذج النمو الأسي.

الكلمات المفتاحية: مسالة القيمة الأولية ، مشكلة التعظيم، مشكلة أسية، مشكلة لوجستية، خوارزمية FLFBA، طرق عددية، خوارزميات Metaheuristic.

Achieved Works

Conferences

- A participation in *Journées des Mathématiques Appliquées (JMA)* by a contribution titled: **An Introduction to Nature Inspired Algorithms for Optimization**. Mohamed Khider University, Biskra, Algeria, 18^{th} and 19^{th}, *Dec*, 2017.

- A participation in *the* 4^{th} *Algerian Congress of Mathematicians (CMA2018)* by a contribution titled: **On the application of the Bio-Inspired Algorithms in Optimization Problems**. M'Hamed BOUGARA University, Boumerdès, Algeria, 12^{th} and 13^{th}, *May*, 2018.

- A participation in *the International Conference on Artificial Intelligence and its Applications (AIAP'18)* by a contribution titled: **Swarm Intelligence Algorithm for Solving Nonlinear Second Order Boundary Value Problems**. Hamma Lakhdar University, El Oued, Algeria, 04^{th} and 05^{th}, *Dec*, 2018.

- A participation in 03^{rd} *International Conference in Operator Theory, PDE and Applications (CITO2019)* by a contribution titled: **Numerical Optimization of Initial Value Problem in Ordinary Differential Equations by Nature inspired Algorithms**. Hamma Lakhdar University, El Oued, Algeria, 24^{th} and 25^{th}, *April*, 2019.

- A participation in *Journée Nationale de Mathématiques Appliquées (JNMA2019)* by a contribution titled: **A Metaheuristic Algorithm for Solving Ordinary Differential Equations of Series RC Circuit - In constant voltage case -**. Larbi Ben M.hidi University, Oum El Bouaghi, Algeria, 27^{th} *June*, 2019.

Publications

- A publication of a paper titled: **A Nature Inspired Algorithm based resolution of an Engineering's ODE**, *International Journal of Scientific Research in*

Mechanical and Materials Engineering (IJSRMME), the Techno Science academy Journals (an international open access publisher), 2018|$Volume$ 2|$Issue$ 2|$ISSN$: 2457 − 0435, (with N. Khelil, 2018).

- A publication of a paper titled: **Solving Initial Value Problems by Flower Pollination Algorithm**, *American Journal of Electrical and Computer Engineering* Science Publishing Group (SciencePG), 2018|$Volume$ 2|$Issue$ 2|$ISSN$: 2640−0502, (with N. Khelil, 2018).

- A publication of a paper titled: **Optimization of Civil Engineering's Initial Value Problems by Particle Swarm Optimization Algorithm**, *International Journal of Scientific Research in Civil Engineering (IJSRCE)*, the Techno Science academy Journals (an international open access publisher), 2019|$Volume$ 3|$Issue$ 1|$Online\ ISSN$: 2456 − 6667, (with N. Khelil, 2019).

- A publication of a paper titled: **Initial Value Problem Resolution Under Optimization Algorithms**, *The Advances and Applications in Mathematical Sciences*, Mili publication,2019|$Volume$ 18|$Issue$ 7|$Pages$ 553 − 565, (with N. Khelil, 2019).

- A publication of a paper titled: **Fractional Lévy Flight Bat Algorithm for Global Optimization**, *International Journal of Bio-Inspired Computation*, the Inderscience publisher,2020|$Volume$ 15|$Issue$ 2|$Pages$ 100 − 112 (with R. Boudjemaa and D. Oliva).

Submissions

- Submission a paper titled: **Modified Salp Swarm Algorithm for Global Optimization**, Neural Computing and Applications, Springer (with R. Boudjemaa).

Table of Acronyms

The different abbreviations and acronyms used throughout this thesis are explained below:

ABM	Adams–Bashforth–Moulton method.
ABC	Artificial Bee Colony.
ANN	Artificial Neural Network.
BA	Bat Algorithm.
BFGS	Broyden-Fletcher-Goldfarb-Shanno Algorithm.
CGP	Cartesian Genetic Programming.
CSA	Crow Search Algorithm.
CS	Cuckoo Search algorithm.
DE	Differential Evolution.
FA	Firefly Algorithm.
FC	Fractional Calculus.
FDPSO	Fractional-order Darwinian Particle Swarm Optimization algorithm.
FLF	Fractional Lévy Flight.
FLFBA	Fractional Lévy Flight Bat Algorithm.
GA	Genetic Algorithm.
GP	Genetic Programming.
GE	Grammatical Evolution.
IWO	Invasive Weed Optimization.
IVP	Initial Value Problem.
LF	Lévy Flight.
MA	Metaheuristic Algorithm.
NFL	No-Free-Lunch theorem.
NBA	Novel Bat Algorithm (with habitat selection and Doppler effects).
ODE	Ordinary Differential Equation.
PDE	Partial Differential Equation.
PSO	Particle Swarm Optimization.
RK4	Runge–Kutta 4^{th} order method.

RK2 Runge–Kutta 2^{nd} order method (Heuns method).

SFS Stochastic Fractal Search.

SI Swarm Intelligence.

WOA Whale Optimization Algorithm.

Contents

List of Figures

List of Tables

General Introduction

Context and Motivations

𝔠*alculus*, originally called infinitesimal calculus, is the mathematical study of continuous change, it has two major branches, differential calculus and integral calculus; the former concerns instantaneous rates of change, and the slopes of curves, while integral calculus concerns accumulation of quantities, and areas under or between curves. These two branches are related to each other by the fundamental theorem of calculus, and they make use of the fundamental notions of convergence of infinite sequences and infinite series to a well-defined limit. One important concept managed in Calculus is *Derivatives* which measures the sensitivity to change of the function value (output value) with respect to a change in its argument (input value). The process of finding a derivative is called *Differentiation*.

In mathematics, a differential equation is an equation that relates one or more functions and their derivatives. In applications, the functions generally represent physical quantities, the derivatives represent their rates of change, and the differential equation defines a relationship between the two. Such relations are common. Mainly the study of differential equations consists of the study of their solutions (the set of functions that satisfy each equation), and of the properties of their solutions. The solution of a differential equation is, in general, an equation expressing the functional dependence of one variable upon one or more others; it ordinarily contains constant terms that are not present in the original differential equations. It produces a function that can be used to predict the behaviour of the original system, at least within certain constraint.

These mathematical entities allow scientists to understand a wide range of complex phe-

nomena. Many fundamental laws of Physics and Chemistry can be formulated as differential equations. They model different problems in diverse scientific fields, such as Biology, Economics or Engineering. Differential equations are mathematically studied from several different perspectives, mostly concerned with their solution, i. e., the set of functions that satisfy the initial differential equation. Only the simplest differential equations are solvable by explicit formulas. When dealing with complex differential equations, an analytic solution becomes difficult to obtain and as a result numerical approximation is sought. The most extended methods to solve differential equations make use of numerical analysis. Several numerical methods have been dedicated to solving Ordinary Differential Equations (ODEs) like Euler method [66], Runge-Kutta method [42], feed forward neural networks [83] and spectral methods like Chebyshev method [70], Legendre methods [30] and optimization algorithms [58, 14]. For that, the equation is discretized into a finite-dimensional subspace. This can be done by a finite element, a finite difference, or a finite volume methods reducing the initial problem to the solution of an algebraic equation. Another class of methods provides an approximation to the analytic solutions, such as the variational iteration method (VIM) [16, 17], the homotopy analysis method (HAM) [23], the method of bilaterally bounded (MBB) [50], and the Adomian double decomposition method (ADM) [18, 4].

Generally, analytically solution for ODEs has some restrictions; the main restriction is that the range of differential equations which should be solved by the analytical method is restricted. Since, in most cases, where the boundary conditions of ODEs are known, a numerical solution can be achieved by approximation methods [38]. There are a lot of numerical methods which are introduced and developed to resolve the considered problem; but, the cases such as fast convergence, reasonably stable and more accuracy are still been researched [12, 48].

One approach to compensate this deficiency is to use the Metaheuristic Algorithms (MAs). Despite there being a wide range of approximate methods for solving ODEs, there is a lack of a proper approach that meets most of the engineering demands having unconventional and nonlinear ODEs. It should be very interesting to solve linear and nonlinear ODEs having arbitrary boundaries and/or initial values. Therefore, when analytical methods

are not capable of solving differential equations or other types of equations in a logical given time, approximation methods are considered as the best solver. Among approximation methods, MAs, devised by observing the phenomena occurring in nature, have demonstrated their capabilities in finding near-optimal solutions to numerical real-valued problems.

MAs perform a mixed of deterministic and stochastic search of the best solution to an optimization problem; *meta-* means 'beyond' or 'higher level', *heuristic* means 'to find' or 'to discover by trial and error' and they generally perform better than simple heuristics. It is worth pointing out that no agreed definitions of heuristics and metaheuristics exist in the literature; some use 'heuristics' and 'metaheuristics' interchangeably. Almost all MAs intend to be suitable for global optimization, it's a way by trial and error to produce acceptable solutions to a complex problem in a reasonably practical time. The complexity of the problem of interest makes it impossible to search every possible solution or combination, the aim is to find good feasible solution in an acceptable timescale. The idea is to have an efficient but practical algorithm that will work most the time and is able to produce good quality solutions. Among the found quality solutions, it is expected some of them are nearly optimal, though there is no guarantee for such optimality. In addition, all MAs use certain tradeoff of randomization and local search, The main mechanism of these methods is the collective behaviour that exists between candidate solutions, which generates a simpler procedure to solve an optimization problem. By searching over a large set of feasible solutions, with the help of its convergence speed and augmentation searched variables number; MAs can often find good solutions with less computational effort than algorithms, iterative methods, or simple heuristics [87].

In the last decades, they have been proposed various MAs to solve complex optimization problems. Some examples are Particle Swarm Optimization (PSO) [41], the Genetic Algorithms (GA) [29] and the Differential Evolution (DE) [78]. In the same context, they have also been proposed novel alternatives like the Whale Optimization Algorithm (WOA) [57], Crow Search Algorithm (CSA) [3], Stochastic Fractal Search (SFS) [75] or the Artificial Bee Colony (ABC) [39]. All these algorithms have been applied to a wide range of applications.

Related Works

Nowadays, applications that use metaheuristic methods for finding approximate solution of ODEs have increased considerably (e.g., GA [56, 29], PSO [50, 41, 5], genetic programming [13]). However, these approaches for solving ODEs are different with each other in terms of applied strategy and base approximate function. For instance, in [50], different strategies named method of bilaterally bounded, has been employed using the PSO. Also, the concept of Fourier series expansion has been used as a base approximate function for finding the approximate solution of ODEs by setting a unit weight function [5]. However, this assumption may not help in obtaining better results for all types of ODEs. Therefore, a new weight function is proposed in [73].

In this thesis we have proposed a new method for solving Initial Value Problems (IVPs) in ODEs based on Fractional Lévy Flight Bat Algorithm (FLFBA) that is a hybridized algorithm which has the nicest characteristics of Bat Algorithm (BA), Lévy Flight (LF) and the Fractional Calculus (FC).

The BA was introduced in 2010 as an alternative method for numerical optimization [88]. BA is based on the mechanism of echolocation in bats, it is a sonar that guides bat along the fly. This behavior also helps bats in hunting, by using the echolocation they can identify the preys in the dark.

The operators of the BA have a good balance between exploration and exploitation that is desirable for MA. However, it has been proved that the performance of BA is good only in problems with a reduced number of dimensions [91], [25]. In this sense, different modifications have been proposed for improving the performance of BA. For example, in [86] is proposed a BA using Differential Evolution (DE) operators and LF during the optimization process. In 2017 it was published a directional BA [20], in which is proposed directional echolocation to improve the exploration of BA. Another interesting improvement was proposed in [25] where the BA is hybridized with DE. The standard BA has also been modified using chaotic maps instead normal distribution to increase the search capabilities of BA [71], [32]. There has also been introduced a modification of BA that considers the GA and the Invasive Weed Optimization (IWO) [91].

FC is an extension of classical mathematics it has been applied in fields like electronics, signal processing, fractals, and chaos [24, 69, 51, 52]. It has also been applied to improve the quality of the solution in modeling, curve fitting or pattern recognition [60]. In this context, the FC is an excellent alternative to introduce concepts as memory (fractional derivative) in different processes; such feature generates more realistic models that integer based models [19]. Besides, the LF has been extensively used to improve different MA [89, 9]. LF can be defined as random walks whose step lengths are not constant, and the values are selected from a probability distribution. It has been demonstrated that LF models the patterns of different species in wildlife [84].

Problematic

Mathematics or particularly applied mathematics is widely used in every engineering fields. It is the background of every engineering domains. Together with physics, mathematics has helped engineering develop. Without mathematics, engineering cannot become so fascinating as it is now.

There is a huge variety of real-life problems optimized by using differential equations; this field is taught as it is important to understand many engineering subjects such as fluid mechanics, heat transfer, electric circuits and mechanics of materials,...etc. When we have to find the optimal solution to a given problem under highly complex constraints.

An IVP is a differential equation:

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}, \tag{1}$$

where t is the independent variable and $y = y(t)$ is the dependent variable, with $f : \Omega \subset \mathbb{R} \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ where Ω is an open set of $\mathbb{R} \times \mathbb{R}^n$, together with a point in the domain of $f(t_0, y_0) \in \Omega$, called the initial condition. Finding the optimal solutions numerically of an IVP is gotten with approximations: $y(t_0 + h), \ldots, y(t_0 + nh)$ where $a = t_0$ and $h = (b - a)/n$. For more precision of the solution, a very small step size h must be used that includes a larger number of steps, thus more computing time which is not available

in the useful numerical methods like Euler or Runge-Kutta methods [35], which may approximate solutions of IVP and perhaps yield useful information, often insufficient in the absence of exact, analytic solutions.

The main aim of our work (problematic) deals with finding a good approximation to IVP by proposing a new MA to optimize numerically an IVP by avoiding the limitations of the analytic solutions, the idea is to create from the conventional BA another MA called FLFBA which is used to improve the convergence to an accurate solution compared to other MAs.

The realization of such approach requires solving several problems:

In order to compensate traditional numerical method's deficiency, new MA inspired from BA called FLFBA is proposed?

That overcomes BA's limitations and permits to optimize the IVP appeared in modeling real life phenomena. This is the global problem when its answer is attached to the resolution of the following sub-questions:

Since BA has nice proprieties, how can we develop a new MA inspired from it?

This is the first contribution when we exploit the strong proprieties of LF and FC that gives a modified version of LF called Fractional Lévy Flight (FLF) to enhance BA.

By using the FLF and DE, BA's performance does it improved?

That is the second contribution. The FLFBA [6] aimed to improve the classical BA and enhance its ability to escape from local optimum. It's tested using several well-known benchmark functions under many advanced nonparametric statistical tests when it's compared with other recent algorithms in terms of convergence and solution quality.

After testing the FLFBA [6], can we integrate it by optimizing real problems envisaged by IVPs?

We consider the IVP in ODEs as an optimization problem. By selecting a specified example which has an important role to describe real problem. The effectiveness of the proposed algorithm is tested via a simulation study.

Does the proposed algorithm lead to a better solution than the traditional numerical methods or other metaheuristics?

FLFBA is a hybridized algorithm which inherits the best characteristics of BA and introduces LF and FC to improve local search routine, global exploration and exploitation of the search space. The performance of FLFBA is tested on IVP and compared to well established numerical methods including Euler methods, Range-Kutta methods, predictor-corrector method, as well as some other MAs like ABCA, BA, FPA.

Thesis Contribution and Objectives

There are the so-called 'No free lunch theorems', which can have significant implications in the field of optimization [85]. One of the theorems states that if algorithm A outperforms than algorithm B for some optimization functions, then B will be superior to A for other functions. In other words, if averaged over all possible function space, both algorithms A and B will perform, on average, equally well. That is to say, there is no universally better algorithms which often involve modification or improvement when applied to a new set of problems because efficiency of an algorithm not guarantee its success.

In this thesis we will present a modified version of the LF using FC that is called FLF, which is the first contribution. The FLF and the DE are used to improve the performance of the standard BA, that is the second contribution. The proposed algorithm is called FLFBA, and it starts by generating a random population of bats positions. The objective function is used to verify the quality of the solutions in the search space.

Two different mechanisms are used in the FLFBA, and they are applied to different sections of the population. The first operator considers two of the best solutions to generate new position in the search space, around such position it is computed a new individual using the LF. The second operator uses the FLF and DE to compute the velocity to move each bat in the population.

The FLFBA is considered as a new hybrid metaheuristics method that combines between proprieties of LF that can maximize the efficiency of resource searches and FC that gives more improvement of the solution quality. This mixture gives more simplicity and flex-

ibility to find local optimum and overcome the lacks found in the traditional BA and keep its good performance in problems with a reduced number of dimensions. By using several well-known benchmark functions and a set of recent algorithms , experimental results and comparisons support the fact that FLFBA is improved regarding accuracy and performance.

The importance of this thesis resides by considering IVPs as an optimization problem implemented by means of FLFBA [6] in order to find numerical solutions for this problem, the obtained results are compared by those of Explicit Euler, Midpoint method, Backward Eulers), Range-Kutta 4^{th} order (RK4) method, Heuns (RK2), Adams–Bashforth–Moulton method (ABM) and those of ABCA, BA, FPA and the exact results of the studied examples. Comparisons are made in terms of solution quality under Matlab software by plotting the numerical results together with the (true) analytical solution +.

Thesis Organization

The remaining chapters are organized as follows:

Part *I* : Preliminary Theory

That includes two chapters:

Chapter 1: **Linear First Order IVP** Section 2 is about the classification of differential equations, Section 3 focus on the solution of differential equations, Section 4 gives some basic definitions about IVP, existence and uniqueness of solutions. Section 5 deals with the most used numerical methods to solve First-Order IVP like Euler method, Runge-Kutta method, etc. Section 6 provides some real-life applications of IVP.

Chapter 2: **Metaheuristics as Optimization Algorithms** The main purpose of this chapter is to provide an overview concerning metaheuristics. Therefore, the chapter is organized as follows: Section 2 furnishs some essential descriptions about optimization algorithms like the parameters of an optimization algorithm. Followed by Section 3 that outlines some details about metaheuristics such as their definition, properties, classification and applications of metaheuristics. Section 4 discusses some metheuristic methods used to solve ODEs. Finally, some discussions are presented in Section 5.

Part *II* : The Main Results

That contains two chapters:

Chapter 3: **Fractional Lévy Flight Bat Algorithm (FLFBA)** Section 2 analyzes the related work such that the basic BA, fractional order, BA equations, Lévy Flight, DE based location update formula. Section 3 presents the proposed FLFBA. Section 4 describes the experimental results based on the parameters settings, benchmark functions and numerical results. Section 5 offers the results analysis supported by the study of different statistical tests: pairwise comparisons, multiple comparisons, post-hoc procedures. Finally, Section 6 discusses the conclusions.

Chapter 4: **Application of FLFBA in Optimizing IVP** This chapter is organized as follows. The formulation of the problem that gives the construction of the objective function and its consistency study are revealed in Section 2. Section 3 provides explication of the population growth models used as application examples in our study, while Section 4 is reserved to the numerical experiments and gives the different parameters setting adopted for FLFBA, ABC, FPA and BA and the parameters adopted for IVP. Then we expose the results that show how FLFBA can lead to a satisfactory result for solving IVP by comparing its performance to a set of numerical methods and MA. The comments and conclusion are made in Section 5.

Part I

Preliminary Theory

Chapter 1

First Order Initial Value Problems

1.1 Introduction

Many problems in natural sciences and engineering fields are formulated into a scalar differential equation or a vector differential equation, that is, a system of differential equations. In this chapter, we look into several methods of obtaining the numerical solutions to ODEs in which all dependent variables (y) depend on a single independent variable (t). ODEs are called an IVP if the values $y(t_0)$ of dependent variables are given at the initial point t_0 of the independent variable. The IVPs will be handled with several methods including Euler method, Runge–Kutta method and predictor–corrector method.

1.2 Classification of Differential Equations

A differential equation is any equation involving derivatives of one or more dependent variables with respect to one or more independent variables. There are many types of differential equations, and a wide variety of solution techniques, even for equations of the same type, let alone different types. We now introduce some terminology that aids in classification of equations and, by extension, selection of solution techniques. The reason is that the techniques for solving differential equations are common to these various classification groups. And sometimes we can transform an equation of one type into an equivalent equation of another type, so that we can use easier solution techniques. Here then are some of the major classifications of differential equations [36] [67]:

On this section we assume that x and y are functions of time, t :

$$x = x(t).$$

$$y = y(t).$$

And the derivatives are with respect to t:

$$\frac{dx}{dt} = \acute{x}(t).$$

1.2.1 Partial vs. Ordinary

- An ordinary differential equation (or ODE) has a discrete (finite) set of variables i.e. when the unknown function y depends on a single independent variable t, then only ordinary derivatives appear in the differential equation. So the equation is called an ODE. For example in the simple pendulum, there are two variables: angle and angular velocity.

- A partial differential equation (or PDE) has an infinite set of variables which correspond to all the positions on a line or a surface or a region of space i.e. when the unknown function y depends on several independent variables $r, s, t, etc.$, partial derivatives appear in the differential equation. So the equation is called a Partial Differential Equations (PDE). For example The heat equation:

$$\frac{\partial u}{\partial t} = k^2 \frac{\partial^2 u}{\partial x^2},$$

  where k is a constant, is an example of a partial differential equation, as its solution $u(x, t)$ is a function of two independent variables, and the equation includes partial derivatives with respect to both variables.

- For an ODE, each variable has a distinct differential equation using "ordinary" derivatives. For a PDE, there is only one "partial" differential equation for each dimension.

- Systems of Differential Equations: We may have two or more dependent variables (unknown functions), then a system of equations is required. For example, predator-

prey equations have the form:

$$\begin{cases} dx/dt = ax - \alpha xy \\ dy/dt = -cy + \gamma xy \end{cases},$$

where $x(t)$ and $y(t)$ are the respective populations of prey and predator species. The constants a, c, α, γ depend on the particular species being studied.

1.2.2 First Order, Second Order

The order of a differential equation is equal to the highest derivative in the equation. The single-quote indicates differention. So $x^{'}$ is a first derivative, while $x^{''}$ is a second derivative.

$x^{'} = 1/x$	is first-order.
$x^{''} = -x$	is second-order.
$x^{''} + 2x^{'} + x = 0$	is second-order.

1.2.3 Linear vs. Nonlinear

An ordinary differential equation:

$$f\left(t, y, y^{'}, y^{''}, y^{'''}, ..., y^{(n)}\right) = 0,$$

is linear if f is linear in y and in its higher derivatives $y^{'}, y^{''}, y^{'''}, ...$ Thus the general linear ODE has the form:

$$a_0(t)y^{(n)} + a_1(t)y^{(n-1)} + ... + a_n(t)y = g(t).$$

A differential equation is linear if any linear combination of solutions of the equation is also a solution of the equation i.e. linear just means that the variable in an equation appears only with a power of one. A differential equation that is not linear is said to be nonlinear. So x is linear but x^2 is nonlinear. Also any function like $\cos(x)$ is nonlinear. In math and physics, linear generally means "simple" and non-linear means "complicated". The theory for solving linear equations is very well developed because linear equations

are simple enough to be solveable. Nonlinear equations are, in general, very difficult to solve and are the subject of much on-going research, so in many cases one approximates a nonlinear equation by a linear equation, called a linearization, that is more readily solved. Linear ODE's enjoy the following three properties:

1. y and all its derivatives are raised to power 1.

2. The coefficients of y and any of its derivatives are functions of t only.

3. No transcendental functions of y and/or its derivatives occur.

Here is a brief description of how to recognize a linear equation. Recall that the equation for a line is:

$$y = mx + b,$$

where m, b are constants (m is the slope, and b is the $y - intercept$). In a differential equation, when the variables and their derivatives are only multiplied by constants, then the equation is linear. The variables and their derivatives must always appear as a simple first power. Here are some examples:

$x'' + x = 0$	is linear.
$x'' + 2x' + x = 0$	is linear.
$x' + 1/x = 0$	is non-linear because $1/x$ is not a first power.
$x' + x2 = 0$	is non-linear becausex^2 is not a first power.
$x'' + sin(x) = 0$	is non-linear because$\sin(x)$ is not a first power.
$xx' = 1$	is non-linear because x' is not multiplied by a constant.

Similar rules apply to multiple variable problems:

$x' + y' = 0$	is linear.
$xy' = 1$	is non-linear because y' is not multiplied by a constant.

Note, however, that an exception is made for the time variable t (the variable that we are differentiating by). We can have any crazy non-linear function of t is not multiplied by a

constant appear in the equation, but still have an equation that is linear in x .

$$\begin{aligned} x'' + 2x' + x &= \sin(t) && \text{is linear in } x. \\ x' + t^2 x &= 0 && \text{is linear in } x. \\ \sin(t)x' + \cos(t)x &= \exp(t) && \text{is linear in } x. \end{aligned}$$

Also, a linear equation can have "constant coefficients" or "variable coefficients". Here are two second order linear ODEs:

$$y''^3 + y' + 5y = 0 \quad y'' - 3ty' + (\cos t)y = 0$$

For nonlinear equations, different terminology is usually used: $y' = e^y$ is called "autonomous", while $y' = e^{ty}$ is called "non-autonomous".

1.2.4 Homogeneous vs. Non-homogeneous

This is another way of classifying differential equations. These fancy terms amount to the following: whether there is a term involving only time, t (shown on the right hand side in equations below).

$$\begin{aligned} x'' + 2x' + x &= 0 && \text{is homogeneous.} \\ x'' + 2x' + x &= \sin(t) && \text{is non-homogeneous.} \\ x' + t^2 x &= 0 && \text{is homogeneous.} \\ x' + t^2 x &= t + t^2 && \text{is non-homogeneous.} \end{aligned}$$

The non-homogeneous part of the equation is the term that involves only time. It usually corresponds to a forcing term in the physical model. For example, in a driven pendulum it would be the motor that is driving the pendulum.

1.3 Solutions to Differential Equations

A solution $\phi(t)$ to an ODE:

$$y^{(n)}(t) = f\left(t, y, y', y'', y''', ..., y^{(n-1)}\right),$$

satisfies the equation:

$$\phi^{(n)}(t) = f\left(t, \phi, \phi', \phi'', \phi''', ..., \phi^{(n-1)}\right).$$

There are three important questions in the study of differential equations:

- Is there a solution? (Existence)
- If there is a solution, is it unique? (Uniqueness)
- If there is a solution, how do we find it? (Analytical Solution, Numerical Approximation, etc)

A differential equation of any type, in conjunction with any other information such as an initial condition, is said to describe a *well-posed problem* if it satisfies three conditions, known as *Hadamard's conditions* for well-posedness:

- A solution of the problem exists.
- A solution of the problem is unique.
- The unique solution depends continuously on the problem data, which may include initial values or coefficients of the differential equation. That is, a small change in the data corresponds to a small change in the solution.

Unfortunately, problems can easily fail to be well-posed by not satisfying any of these conditions. However, in this thesis we deal with IVP that are well-posed. Now here are some rules of thumb for when we can solve the ODE (meaning we obtain a specific formula for $y(t)$, or at least an equation defining $y(t)$ implicitly), and when we can't:

1. All first and second order linear equations with constant coefficients can be solved.
2. All first order linear equations can be solved, at least in terms of an integral.
3. Second or higher order linear ODEs with variable coefficients usually cannot be solved.
4. Third or higher order linear ODE with constant coefficients can be solved occasionally. The difficulty in solving them is doing the algebra.

5. First order nonlinear ODEs can sometimes be solved. For example, separation of variables might work.

6. Second and higher order nonlinear ODE can rarely be solved.

1.4 Initial Value Problems (IVPs)

In the field of differential equations, an IVP (also called a Cauchy problem by some authors) is an ordinary differential equation together with a specified value, called the initial condition, of the unknown function at a given point in the domain of the solution. In physics or other sciences, modeling a system frequently amounts to solving an IVP; in this context, the differential initial value is an equation that is an evolution equation specifying how, given initial conditions, the system will evolve with time.

1.4.1 Definition of IVP

An IVP is a differential equation

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases}, \tag{1.1}$$

where t is the independent variable and $y = y(t)$ is the dependent variable, with $f : \Omega \subset \mathbb{R} \times \mathbb{R}^n \longrightarrow \mathbb{R}^n$ where Ω is an open set of $\mathbb{R} \times \mathbb{R}^n$, together with a point in the domain of $f(t_0, y_0) \in \Omega$, called the initial condition. A solution to an IVP is a function y that is a solution to the differential equation and satisfies:

$$y(t_0) = y_0.$$

More generally, the unknown function y can take values on infinite dimensional spaces, such as Banach spaces or spaces of distributions [36]. IVP are extended to higher orders by treating the derivatives in the same way as an independent function, e.g.

$$y''(t) = f(t, y(t), y'(t)).$$

Most differential equations have more than one solution. For a first-order equation, the general solution usually involves an arbitrary constant C, with one particular solution corresponding to each value of C. What this means is that knowing a differential equation that a function $y(t)$ satisfies is not enough information to determine $y(t)$. To find the formula for $y(t)$ precisely, we need one more piece of information, usually called an initial condition. In general, we expect that every IVP has exactly one solution. We can find this solution using the following procedure.

1. Find the general solution to the given differential equation, involving an arbitrary constant C.
2. Substitute $t = t_0$ and $y = y_0$ to get an equation for C.
3. Solve for C and then substitute the answer back into the formula for y.

1.4.2 Existence and uniqueness of solutions

As a general rule, we expect any IVP of the form Eq. (1.1) to have a unique solution. The following theorem gives specific conditions which guarantee that this hold.

Theorem 1.1 (Existence and uniqueness of solutions) *Consider an IVP of the form Eq. (1.1). If the function $f(t, y)$ is continuously differentiable [1] for all values of t and y, then this IVP has a unique solution.*

Since this theorem is also known as the existence and uniqueness theorem for first order ODEs, it guarantees both that the solution exists and that it is unique. The hypothesis that the function $f(t, y)$ is continuously differentiable is important for the theorem. In fact, there are IVP that does not satisfy this hypothesis that has more than one solution. For example, the IVP:

$$\begin{cases} y' = \dfrac{y}{t'} \\ y(0) = 0 \end{cases},$$

has infinitely many different solutions, namely the lines $y = Ct$ for all possible values of C. The function $f(t, y)$ in this case is y/t, which is not defined (and hence not continuously

[1] Here continuously differentiable means that both partial derivatives $\dfrac{\partial f}{\partial t}$ and $\dfrac{\partial f}{\partial y}$ exist and are continuous.

differentiable) when $t = 0$. There is a nice geometric interpretation of the fundamental theorem. As we have seen, the solutions to a differential equation can be viewed as a family of solution curves in the $ty - plane$. From a geometric point of view an initial condition $y(a) = b$ is the same as a point (a, b) that the solution curve must pass through. Thus, saying that the IVP (Eq. 1.1) has a unique solution is the same as saying that the point (a, b) has exactly one solution curve passing through it. This leads us to the following restatement of the fundamental theorem of ODEs.

Theorem 1.2 (Existence and uniqueness of solutions (Geometric Version)) *Consider a first-order differential equation of the form $y' = f(t, y)$, where the function $f(t, y)$ is continuously differentiable. Then:*

1. The solution curves for this differential equation completely fill the plane,

2. Solution curves for different solutions do not intersect.

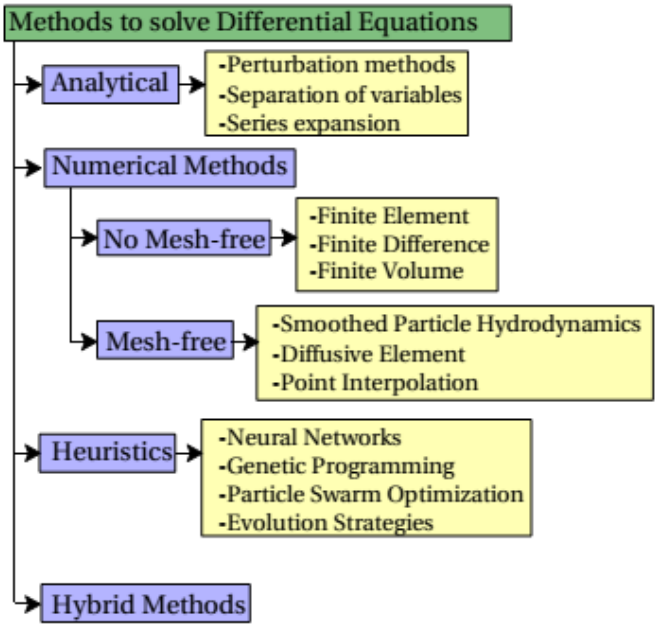
Here statement (1) is the same as saying that every point (a, b) lies on at least one solution curve, i.e. every initial condition gives at least one solution. Statement (2) is the same as saying that no point (a, b) lies on more than one solution curve, i.e. every initial condition has at most one solution.

1.5 Numerical Solutions of First-Order IVP

Some simple differential equations admit solutions given by explicit formulas. But in the general case, only approximated solutions can be found. Several paradigms exist in the literature to solve the equations. Figure (1.1) shows a possible taxonomy of existing methods to solve differential equations [15, 44].

For most differential equations it is impossible to write down a solution formula using elementary functions. Even when such a formula exists, it might be difficult to draw any conclusions from it. Numerical methods are therefore an indispensable tool for studying differential equations, especially when combined with qualitative methods. These notes are meant to serve as a very brief introduction to numerical methods for ODEs. If you wish to find out more, you can have a look in the list of references. We will ignore many things related to the practical implementation of these methods on a computer. In

Figure 1.1: Methods to solve differential equations.



particular, we will not discuss the speed of the various algorithms (roughly proportional to the number of operations involved). We simply note that lowering the step size increases the accuracy, but also the number of operations. Methods with high accuracy at relatively large step sizes are therefore preferable. On the other hand, such methods usually involve many computations in each step. We will also ignore round-of errors. Computers have finite precision and it is therefore not possible in practice to make the error arbitrarily small by shrinking the step size. Moreover, when the step size is very small, round-off errors become dominant and the error analysis presented in these notes becomes invalid [15, 35, 44].

1.5.1 Euler method

To illustrate the ideas, we consider Euler's method. For simplicity, we shall assume that the initial time is $t_0 = 0$ and we will only be concerned with what happens for $t \geq 0$. In order to simplify the discussion, we will also concentrate on first-order equations. Many of the results in the notes can however be generalized to systems and hence to higher-order equations. We wish to solve the IVP in Eq. (1.1). In Euler's method, we discretize time

by setting $t_k = hk$, $k = 0, 1, 2, ...$, and define an approximate solution by:

$$x_{k+1} = x_k + hf_k, \tag{1.2}$$

where $f_k = f(t_k, x_k)$. When we wish to emphasize the dependence on the step size, we write $x_k^{(h)}$. One way of motivating Euler's method is to approximate the derivative with a finite difference:

$$\acute{x}(t) \approx \frac{x(t+h) - x(t)}{h}.$$

Solving for $x(t+h)$ gives:

$$x(t+h) \approx x(t) + h\acute{x}(t) = x(t) + hf(t, x(t)).$$

Here we assume that $h > 0$, so that $x(t+h)$ can be computed from $x(t)$. In many cases, one considers Eq. (1.1) on a fixed time interval $[0, T]$. It is then natural to choose $h = T/N$ for some integer N. By letting $N \to \infty$ we can make $h \to 0$. Eq. (1.2) defines the approximate solution only at the discrete points t_k. One can extend the approximate solution to all t by using linear interpolation between t_k and t_{k+1}:

$$x^{(h)}(t) = \frac{t - t_k}{t_{k+1} - t_k}(x_{k+1}^{(h)} - x_k^h) + x_k^{(h)}, \qquad t \in [t_k, t_{k+1}].$$

This is particularly useful if one wants to interpret the method geometrically. The exact solution of the Eq. (1.1) is tangent to the direction field $(1, f(t, x))$ for all values of t, whereas the approximate solution is a polygon in which the slopes are given by the direction field at the points t_k.

Error estimates and convergence: A minimal requirement of a numerical method is that it converges to a solution of the associated IVP as the step size $h \to 0$. We will now show that this is the case for Euler's method under reasonable assumptions on f. We begin by defining two different ways of measuring the error between the numerical solution and the exact solution [44, 74].

Definition 1.1 *The local truncation error* T_{k+1} *in step* $k+1$ *is the difference between* x_{k+1} *and* $x(t_{k+1})$*, assuming that the numerical solution equals the exact solution at step*

k. In other words,

$$T_{k+1} = \tilde{x}_{k+1} - x(t_{k+1}),$$

where $\tilde{x}_{k+1} = x(t_k) + hf(t_k, x(t_k))$.

The local truncation error is the error produced in one step of the method, assuming that the input data is exact. When applying the method repeatedly, errors will accumulate.

Definition 1.2 *The global truncation error* $E_{k+1} = x_{k+1} - x(t_{k+1})$ *in step* $k+1$ *is the difference between* x_{k+1} *and* $x(t_{k+1})$.

Throughout the following discussion, we assume that f and its derivatives are bounded in the domain in which the exact and approximate solutions are defined. We let M denote a generic constant depending on the maximum of f and its derivatives. The exact value may vary from line to line.

Proposition 1.1 *The local truncation error for Euler's method satisfies* $|T_{k+1}| \leq Mh^2$.

Using Landau notation, we can write this as $T_{k+1} = O(h^2)$ as $h \to 0$. In general, a method satisfying $T_{k+1} = O(h^{p+1})$ is said to be of order p. The above proposition doesn't really tell us anything about the convergence of the approximate solution to the exact solution. For this we need to estimate the global truncation error. We consider now a fixed time interval $[0, T]$.

Proposition 1.2 *The global truncation error for Euler's method satisfies* $E_{k+1} = O(h)$.

In fact, this estimate is uniform over $[0, T]$:

$$E(h) := \max_{0 \leq k \leq N} |E_k| = O(h), \qquad h = T/N.$$

In particular this means that the approximate solution converges to the exact solution as $h \to 0$.

1.5.2 Implicit methods

In the derivation of Euler's method we replaced the derivative with a forward difference quotient. If one instead uses a backward difference quotient:

$$\acute{x}(t) \approx \frac{x(t) - x(t-h)}{h}, \qquad h > 0,$$

and solves for $x(t)$ one obtains the backward Euler method:

$$x_{k+1} = x_k + hf_{k+1}. \tag{1.3}$$

This is an example of an implicit method. Note that f_{k+1} depends on x_{k+1}, so that x_{k+1} is only implicitly determined by x_k. Only in rare cases is it possible to solve this equation explicitly. Normally one has to solve it using some numerical method, e.g. using fixed point iteration:

$$x_{k+1}^{(j+1)} = x_k + hf(t_{k+1}, x_{k+1}^{(j)}), \qquad j = 0, 1, 2....$$

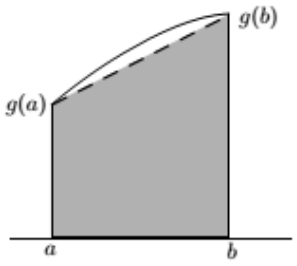
Under suitable conditions on h and $x_{k+1}^{(0)}$, this will converge to a solution x_{k+1} as $j \to \infty$. Doing many iterations is, however, computationally costly. Nevertheless, implicit methods are sometimes used since they generally have better stability properties than explicit methods. Implicit methods are particularly useful when combined with an explicit method to calculate the starting value $x_{k+1}^{(0)}$. One then obtains a so-called predictor-corrector method. The implicit method 'corrects' the explicit method (the 'predictor'). It is then often enough to do a small number of fixed point iterations.

1.5.3 Higher-order methods

A natural way of interpreting Euler's method, which lends itself well to generalizations, is to regard it as an approximation of integrals. If we integrate the differential equation between t and $t + h$, we find that [15, 74]:

$$x(t+h) = x(t) + \int_t^{t+h} f(s, x(s))ds.$$

Figure 1.2: The trapezoidal rule.



This integral can be approximated by:

$$\int_t^{t+h} f(s, x(s))ds = f(\tau, x(\tau))h + O(h^2),$$

where $\tau \in [t, t+h]$ is arbitrary. The choice $\tau = t$ results in the explicit Euler method, while $\tau = t + h$ gives the implicit one. In order to obtain better methods, one can approximate the integral in other ways. The trapezoidal rule [15, 35]:

$$\int_a^b g(x)dx \approx \frac{b-a}{2}(g(a) + g(b)),$$

suggests the scheme:

$$x_{k+1} = x_k + \frac{h}{2}(f_k + f_{k+1}),$$

known simply as the trapezoidal method (see Figure 1.2). This is an implicit scheme. When f is independent of x, the scheme coincides with the trapezoidal rule. If we replace x_{k+1} in f_{k+1} by the approximate value $x_k + hf_k$, we obtain the explicit method

$$x_{k+1} = x_k + \frac{h}{2}(f_k + f(t_{k+1}, x_k + hf_k)),$$

known as Heun's method. One can show that both of these methods are second order (with $E(h) = O(h^2)$). The trapezoidal method and Heun's method are examples of Runge-Kutta methods. The general form of a Runge-Kutta method is:

$$x_{k+1} = x_k + h\sum_{i=1}^{p} b_i r_i,$$

where

$$r_i = f\left(t_k + c_i h, x_k + h\sum_{i=1}^{p} a_{ij} r_j\right), \qquad i = 1, 2, ..., p,$$

with certain conditions on the coefficients b_i, c_i and a_{ij}. The method is explicit if $a_{ij} = 0$ for $i \leq j$ and otherwise implicit. The most famous example of a Runge-Kutta method is probably the following fourth-order version: [2].

$$x_{k+1} = x_k + \frac{h}{6}(r_1 + 2r_2 + 2r_3 + r_4),$$

where

$$r_1 = f_k, \qquad r_2 = f\left(t_k + \tfrac{h}{2}, x_k + \tfrac{h}{2} r_1\right),$$
$$r_3 = f\left(t_k + \tfrac{h}{2}, x_k + \tfrac{h}{2} r_2\right), \qquad r_4 = f(t_k + h, x_k + h r_3).$$

We will not discuss how one arrives at these precise values. We simply note that if f is independent of x, the method coincides with Simpson's rule for approximating integrals [3]:

$$\int_a^b g(x)dx \approx \frac{b-a}{6}\left(g(a) + 4g\left(\frac{a+b}{2}\right) + g(b)\right).$$

The equation is $\acute{x} = x$ with $x(0) = 1$. The number of steps is 2^k for $k = 0, 1, ..., 13$ and the interval of integration $[0, 1]$. The global error E_N is plotted for each N. Note that this means that we are comparing at the fixed time $t = 1$. The scale is double logarithmic. This makes sense since we expect that:

$$E_N = c.h^p = c.N^{-p},$$

where p is the order of the method. Hence,

$$\log E_N = \log c - p \log N$$

gives a straight line with slope $(-p)$. Note that the Runge-Kutta method starts to behave strangely when $E_N \approx 10^{-15}$. This is due to round-off errors. The exact limit depends on

[2] This is usually what is meant if someone says 'Runge-Kutta' without further qualification.

[3] Simpson's rule is obtained by approximating the integrand with a second-degree polynomial.

the fact that we are using double precision.

1.5.4 Multistep methods

The methods discussed so far have all been one-step methods, meaning that x_{k+1} is calculated only using x_k, without taking into account the previous values $x_0, x_1, ..., x_{k-1}$. The idea behind multistep methods is to also use these previous values. A linear multistep method has the form [7, 74]:

$$x_{k+1} = \sum_{i=0}^{p-1} a_j x_{k-j} + h \sum_{i=-1}^{p-1} b_j f_{k-j}, \qquad k = p, ..., N-1. \tag{1.4}$$

If we assume that $(a_{p-1}, b_{p-1}) \neq (0, 0)$ this is a $p - step$ method. The method is explicit if $b_{-1} = 0$ (so that f_{k+1} doesn't appear), otherwise it is implicit. Eq.(1.4) is a difference equation. Except for special functions f, it is difficult to say anything general about the solution of the difference equation. As we will see in the examples below, the difference equation may have properties which differ a lot from the differential equation it is supposed to approximate. So far we haven't said anything about how to choose the coefficients a_j and b_j. A first basic requirement is that the difference equation should be consistent with the differential equation. This basically means that Eq. (1.4) approximates Eq. (1.1) and not some other equation. The precise definition is that $T_k/h \to 0$ as $h \to 0$. It is possible to translate this into a condition on the coefficients, but we will not go into details here. The examples that we consider are consistent. If we approximate $\acute{x}(t)$ with the symmetric difference:

$$\frac{x(t+h) - x(t-h)}{2h},$$

we are led to the two-step recursion formula:

$$x_{k+1} = x_{k-1} + 2hf_k, \qquad k = 1, ..., N-1.$$

This is the midpoint method. The recursion formula requires two initial values x_0 and x_1. One can show that the local truncation error is $O(h^3)$ and that the global error is $O(h^2)$. In particular, the method is consistent. Even though the exact solution converges

to 0, the approximate solution grows without bound [4]. Note that this is a property of the numerical method rather than the differential equation. In fact, it results from the fact that we approximate a first-order differential equation with a second-order difference equation. Fortunately, not all multistep methods are unstable.

Example 1.1 *The Adams-Bashforth methods are a famous family of multistep methods. There is one method for each value of $p \geq 1$. For $p = 1$, it's simply Euler's method. For $p = 2$, the method takes the form:*

$$x_{k+1} = x_k + \frac{h}{2}(3f_k - f_{k-1}),$$

and in general it has the form:

$$x_{k+1} = x_k + h\sum_{i=1}^{p-1} b_j f_{k-j},$$

for some coefficients b_j.

1.6 Real-life applications of IVP

These are a variety of applications of first order differential equations to real world systems:

1. Cooling/Warming Law: the mathematical formulation of Newton's empirical law of cooling of an object in given by the linear first-order differential equation.

2. Population Growth and Decay: $dN(t)/dt = kN(t)$ where $N(t)$ denotes population at time t and k is a constant of proportionality, serves as a model for population.

3. Radio Active Decay and Carbon Dating: a radioactive substance decomposes at a rate proportional to its mass. This rate is called the decay rate. If $m(t)$ represents the mass of a substance at any time, then the decay rate dm/dt is proportional to $m(t)$. Let us recall that the half-life of a substance is the amount of time for it to decay to one half of its initial mass.

[4] In fact, there are many examples where the instability prevents the approximate solution from converging to the exact solution, even though the method is consistent.

Equation	Expression
Newton second law	$F = \frac{d(mv)}{dt}$
Wave	$\frac{\partial^2 u}{\partial^2 t^2} = c^2 \nabla^2 u$
Laplace	$\nabla^2 \varphi = 0$
Heat	$\frac{\partial u}{\partial t} = \alpha \nabla^2 u$
Poisson	$\nabla^2 \varphi = f$
Malthusian growth model	$\frac{dx}{dt} = kx$
Verhust	$\frac{d}{dx} f(x) = f(x).(1 - f(x))$

Table 1.1: Some famous differential equations in physics, engineering, biology and economics.

4. Series Circuits: There are many applications of differential equations indeed like the resistive capacitive circuits can be analyzed using 1^{st} order equations. This helps us to find charges stored in capacitors and current in circuit at any time t.

5. There are wide application of ODE in recent times:

   (a) Cancer/AIDS growth and chemotherapy modeling

   (b) Epidemic disease modeling e.g. dengue, HFMD, malaria...

   (c) Rumor/malware models in social electronic network and Security/Terrorism related models

To conclude this section, some famous differential equations are enumerated in Table (1.1), they are related to physics, engineering, biology and economics.

Chapter 2

Metaheuristics As Optimization Algorithms

2.1 Introduction

Real-world optimization problems are often very challenging to solve, and many applications have to deal with hard problems. To solve such problems, optimization tools have to be used, though there is no guarantee that the optimal solution can be obtained. In fact, for complex problems, there are no efficient algorithms at all. As a result, many problems have to be solved by trial and errors using various optimization techniques. In addition, new algorithms have been developed to see if they can cope with these challenging optimization problems. Among these new algorithms, many metaheuristics, have gained popularity due to their high efficiency [8, 87].

2.2 Optimization

It is no exaggeration to say that optimization is everywhere, from engineering design to business planning and from the routing of the Internet to holiday planning. In almost all these activities, we are trying to achieve certain objectives or to optimize something such as profit, quality and time. As resources, time and money are always limited in real-world applications, we have to find solutions to optimally use these valuable resources under various constraints. Mathematical optimization or programming is the study of such

planning and design problems using mathematical tools. Nowadays, computer simulations become an indispensable tool for solving such optimization problems with various efficient search algorithms [87].

2.2.1 Definition

Mathematically speaking, it is possible to write most optimization problems in the generic form [11]:

$$\begin{aligned} &\underset{x\in\Re^n}{\text{minimize}} f_i(x), && (i=1,2,...,M), \\ &\text{subject to } h_j(x)=0, && (j=1,2,...,J), \\ &g_k(x)\leq 0, && (k=1,2,...,K), \end{aligned}$$

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

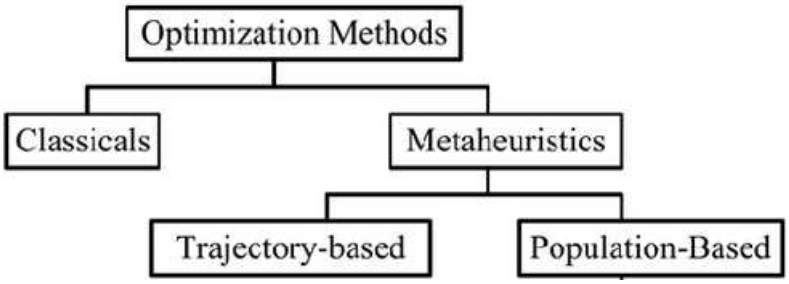
$$x=(x_1,x_2,...,x_n)^T.$$

Here the components x_i of x are called *design* or *decision variables*, and they can be real continuous, discrete or the mixed of these two. The functions $f_i(x)$ where $i=1,2,...,M$ are called the *objective functions* or simply *cost functions*, and in the case of $M=1$, there is only a single objective. The space spanned by the decision variables is called the *design space* or *search space* $\Re^n$, while the space formed by the objective function values is called the *solution space* or *response space*. The equalities for h_j and inequalities for g_k are called *constraints*. It is worth pointing out that we can also write the inequalities in the other way ≥ 0, and we can also formulate the objectives as a maximization problem. In a rare but extreme case where there is no objective at all, there are only constraints. Such a problem is called a *feasibility problem* because any feasible solution is an optimal solution [87].

Optimization problems can be classified according to the number of objectives into two categories:

- Single objective optimization: if $M=1$.
- Multiobjective optimization: if $M>1$, is also referred to as multicriteria or even multi-attributes optimization in the literature. In real-world problems, most optimization

Figure 2.1: Optimization methods.



tasks are multiobjective.

Similarly, we can also classify optimization in terms of number of constraints $J + K$:

- Unconstrained optimization problem: if there is no constraint at all $J = K = 0$.
- Equality-constrained problem: if $K = 0$ and $J \geq 1$.
- Inequality-constrained problem: if $J = 0$ and $K \geq 1$. It is worth pointing out that in some formulations in the optimization literature, equalities are not explicitly included, and only inequalities are included. This is because an equality can be written as two inequalities. For example $h(x) = 0$ is equivalent to $h(x) \leq 0$ and $h(x) \geq 0$.

We can also use the actual function of the objective functions forms for classification:

- Linearly constrained problem: if the constraints h_j and g_k are all linear.
- Linear programming problem: if both the constraints and the objective functions are all linear, here 'programming' has nothing to do with computing programming, it means planning and/or optimization.
- Nonlinear optimization problem: if all f_i, h_j and g_k are nonlinear.

2.2.2 Search for optimality

After an optimization problem is formulated correctly, the main task is to find the optimal solutions by some solution procedure using the right mathematical techniques. The most likely scenario is that we will do a random walk. Such random walk is a main characteristic

of modern search algorithms, so the whole path is a trajectory-based search. Alternatively, we can use the so-called swarm intelligence as we see in almost all modern metaheuristic algorithms, we try to use the best solutions or agents, and randomize (or replace) the not-so-good ones, while evaluating each individual's competence (fitness) in combination with the system history (use of memory). With such a balance, we intend to design better and efficient optimization algorithms [79, 27].

Classification of optimization algorithm can be carried out in many ways. A simple way is to look at the nature of the algorithm, and this divides the algorithms into two categories (see Figure 2.2) [87]:

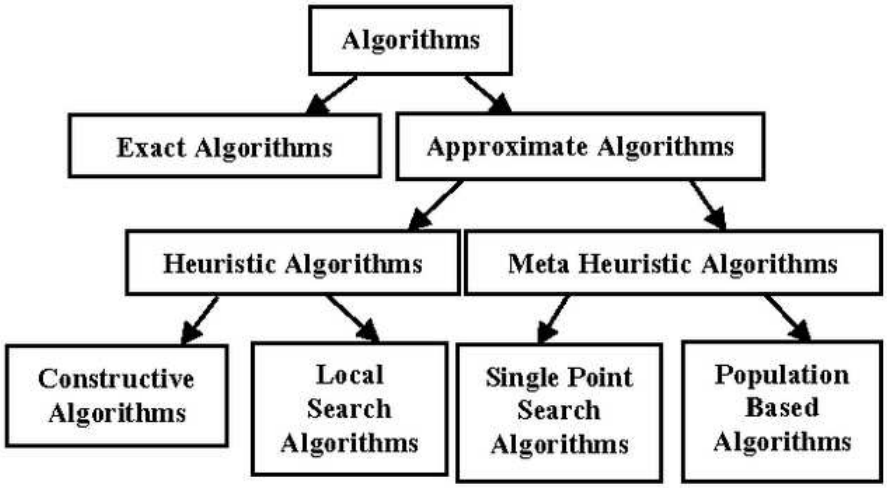
- Deterministic algorithms: they follow a rigorous procedure, and its path and values of both design variables and the functions are repeatable
- Stochastic algorithms: they always have some randomness. Genetic algorithms are a good example, the strings or solutions in the population will be different each time you run a program since the algorithms use some pseudo-random numbers, though the final results may be no big difference, but the paths of each individual are not exactly repeatable.
- Mixture, or a hybrid algorithm, of deterministic and stochastic algorithms: The basic idea is to use the deterministic algorithm, but start with different initial points. However, since there is a random component in this hybrid algorithm, we often classify it as a type of stochastic algorithm in the optimization literature.

2.2.3 Optimization algorithms

Definition 2.1 *An optimization algorithm (see Figure 2.1) is essentially an iterative procedure, starting with some initial guess point/solution with an aim to reach a better solution or ideally the optimal solution to a problem of interest.*

This process of search for optimality is generic, though the details of the process can vary from algorithm to algorithm. Traditional algorithms such as Newton-Raphson methods use a deterministic trajectory-based method, while modern nature-inspired algorithms often are population-based algorithms that use multiple agents. In essence, these multiple

Figure 2.2: Classification of algorithms.



agents form an iterative, dynamic system which should have some attractors or stable states. On the other hand, the same system can be considered as a set of Markov chains so that they will converge toward some stable probability distribution [77, 79].

2.2.4 Parameters of an Optimization Algorithm

1. **Initial approximation:** To initialize the algorithm, it is necessary to have an initial approximation to the solution x_0. (Starting point). The choice of a good initial approximation conditions the convergence or not to the solution.

2. **Number of iterations:** An optimization algorithm uses a recursive process, calculates a new approximation (iteration) to the actual solution until the convergence criteria are reached. In programming, it's a rehearsal loop where the new approximation is constructed from previous approximations.

3. **Convergence speed:** When we talk about convergence close to a solution, we talk about the speed with which terms of iterations are approaching its limit.

$$\lim_{n \longrightarrow \infty} \frac{|x_{n+1} - \xi|}{|x_n - \xi|^q} = \mu, \text{with } \mu > 0 \text{ and } q \text{ is the convergence order.}$$

In general, the orders of convergence are linear ($q = 1$), quadratic ($q = 2$), cubic ($q = 3$), quartic ($q = 4$)..., etc. An optimization method with a convergence order

superior arrives at the solution with few iterations. Choosing a method with a high convergence is important for problems of a certain size or with multiple settings. For example, for a quadratic convergence, we can say that the number of correct digits are double (at least) at each computation step. Or say under another form, the error decreases quadratically at each iteration. If an algorithm does not converge, that does not mean that there is no solution. There exists no universal algorithm whose convergence is guaranteed, in general it depends on the choice of the initialization x_0 and the properties of the function (continuity, differentiability).

4. **Stopping criterion:** Criteria to stop the calculation process. There are several criteria for stopping. Most used:

   (a) Maximum number of iterations N_{max}.

   (b) $\|f(x_n)\| < \varepsilon_1$ function value.

   (c) $\|x_{n+1} - x_n\| < \varepsilon_2$ difference between two successive approximations. Where $\varepsilon_1, \varepsilon_2 \in \mathbb{R}$ are the tolerances and are chosen according to the type of problem. In general, these are negligible values ($\varepsilon_i \approx 10^{-4} - 10^{-6}$) [87].

2.3 Metaheuristics

2.3.1 Definition of metaheuristics

In computer science and mathematical optimization, a metaheuristic is a higher-level procedure or heuristic designed to find, generate, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem, especially with incomplete or imperfect information or limited computation capacity [11].

Metaheuristics sample a set of solutions which is too large to be completely sampled. Metaheuristics may make few assumptions about the optimization problem being solved, and so they may be usable for a variety of problems [8, 77].

Compared to optimization algorithms and iterative methods, metaheuristics do not guarantee that a globally optimal solution can be found on some class of problems [10]. Many metaheuristics implement some form of stochastic optimization, so that the solution found

is dependent on the set of random variables generated. In combinatorial optimization, by searching over a large set of feasible solutions, metaheuristics can often find good solutions with less computational effort than optimization algorithms, iterative methods, or simple heuristics. As such, they are useful approaches for optimization problems [8, 11].
Several books and survey papers have been published on the subject [87, 79, 27]. Most literature on metaheuristics is experimental in nature, describing empirical results based on computer experiments with the algorithms. But some formal theoretical results are also available, often on convergence and the possibility of finding the global optimum [11]. Many metaheuristic methods have been published with claims of novelty and practical efficacy. While the field also features high-quality research, many of the publications have been of poor quality; flaws include vagueness, lack of conceptual elaboration, poor experiments, and ignorance of previous literature.

2.3.2 Properties of metaheuristics

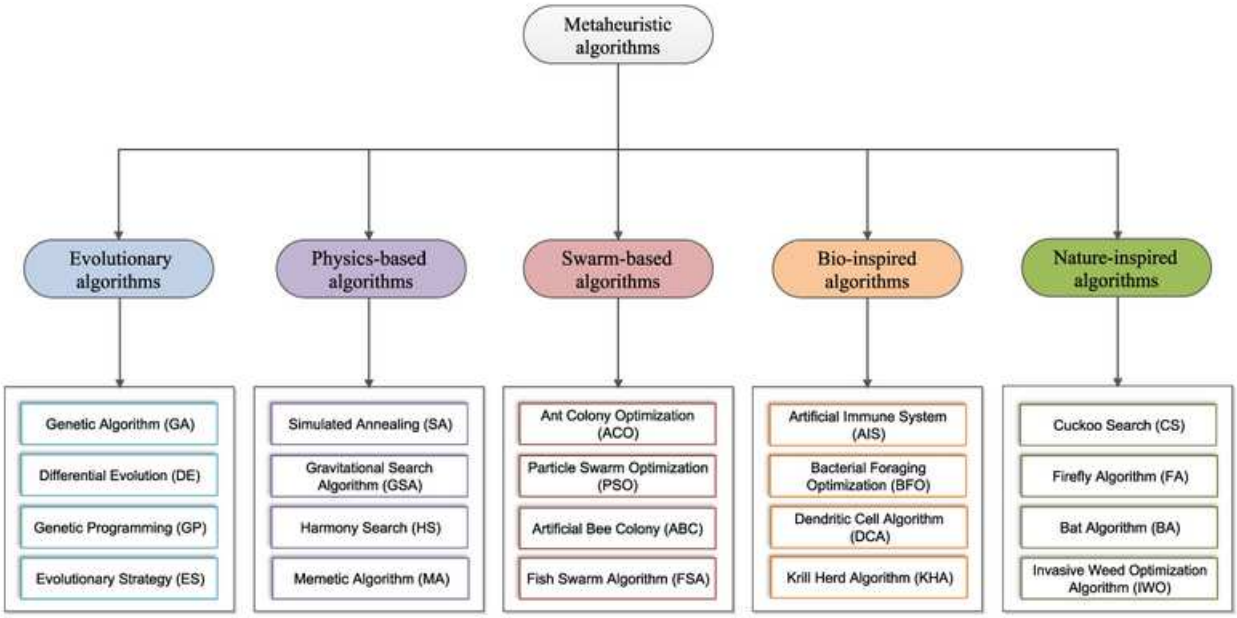
These are properties that characterize most metaheuristics [10]:

- Metaheuristics are strategies that guide the search process.
- The goal is to efficiently explore the search space in order to find near–optimal solutions.
- Techniques which constitute metaheuristic algorithms range from simple local search procedures to complex learning processes.
- Metaheuristic algorithms are approximate and usually non-deterministic.
- Metaheuristics are not problem-specific.

2.3.3 Classification of metaheuristics

There are a wide variety of metaheuristics and a number of properties with respect to how classify them [10, 11]:

Figure 2.3: Classification of metaheuristic algorithms.



Local search vs. global search

One approach is to characterize the type of search strategy. One type of search strategy is an improvement on simple local search algorithms. Many metaheuristic ideas were proposed to improve local search heuristics in order to find better solutions. Such metaheuristics include simulated annealing, tabu search, iterated local search, variable neighborhood search. These metaheuristics can both be classified as local search-based or global search metaheuristics. Other global search metaheuristic that are not local search-based are usually population-based metaheuristics. Such metaheuristics include ant colony optimization, evolutionary computation, particle swarm optimization, and genetic algorithms [10].

Single-solution vs. Population-based

Another classification dimension is single solution vs. population-based searches. Single solution approaches focus on modifying and improving a single candidate solution; single solution metaheuristics include simulated annealing, iterated local search, variable neighborhood search, and guided local search [10, 79].

Population-based approaches maintain and improve multiple candidate solutions, often

Figure 2.4: Swarm intelligence

(a) Bee's swarm

(b) Bird's swarm

(c) Ant's swarm

using population characteristics to guide the search; population based metaheuristics include evolutionary computation, genetic algorithms, and particle swarm optimization. Another category of metaheuristics is Swarm intelligence which is a collective behavior of decentralized, self-organized agents in a population or swarm. Ant colony optimization, particle swarm optimization, social cognitive optimization, penguins search optimization algorithm and artificial bee colony algorithms are examples of this category [33, 21, 79] (Figure 2.4).

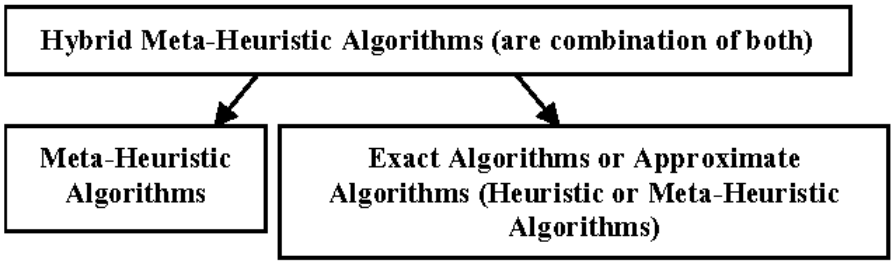
Hybridization and Mimetic algorithms

A hybrid metaheuristic is one which combines a metaheuristic with other optimization approaches, such as algorithms from mathematical programming, constraint programming, and machine learning. Both components of a hybrid metaheuristic may run concurrently and exchange information to guide the search. On the other hand, mimetic algorithms [39] represents the synergy of evolutionary or any population-based approach with separate individual learning or local improvement procedures for problem search. An example of mimetic algorithm is the use of a local search algorithm instead of a basic mutation operator in evolutionary algorithms (see Figure 2.5).

Nature-inspired metaheuristics

A very active area of research is the design of nature-inspired metaheuristics. Many recent metaheuristics, especially evolutionary computation-based algorithms, are inspired by natural systems. Nature acts as a source of concepts, mechanisms and principles for designing of artificial computing systems to deal with complex computational problems

Figure 2.5: Hybrid metaheuristic algorithms.



[33]. Such metaheuristics include simulated annealing, evolutionary algorithms, ant colony optimization and particle swarm optimization. A large number of more recent metaphor-inspired metaheuristics have started to attract criticism in the research community for hiding their lack of novelty behind an elaborate metaphor.

2.3.4 Applications of Metaheuristics

In solving optimization problems, traditional optimization methods such as gradient-based methods may not be able to cope with high non-linearity and multi-modality. Metaheuristic algorithms tend to produce better results for highly nonlinear problems:

1. Metaheuristics are used for combinatorial optimization in which an optimal solution is sought over a discrete search space. An example problem is the traveling salesman problem where the search space of candidate solutions grows faster than exponentially as the size of the problem increases, which makes an exhaustive search for the optimal solution infeasible.

2. Additionally, multidimensional combinatorial problems, including most design problems in engineering [80, 26] such as form-finding and behavior-finding, suffer from the curse of dimensionality, which also makes them infeasible for exhaustive search or analytical methods.

3. Metaheuristics are also widely used for job shop scheduling and job selection problems [92].

4. Popular metaheuristics for combinatorial problems include simulated annealing by Kirkpatrick et al.[45], genetic algorithms by Holland et al. [34], scatter search and

Algorithm	Inspiration
Genetic algorithm	Darwinian evolution in nature
Simulated annealing	Annealing process of materials
Ant colony optimization	Behavior of ants foraging
Bee algorithm	Behavior of bees
Particle swarm optimization	Swarming behavior of birds and fish
Tabu search	Human memory
Harmony search	Musical performance
Big Bang big crunch	Evolution of the universe
Firefly algorithm	Flashing characteristic of fireflies
Cuckoo search	Brood parasitic behavior of cuckoo species
Charged system search	Electrostatic and Newtonian mechanic laws
Bat algorithm	Echolocation characteristic of bats
Eagle strategy	Foraging behavior of eagles
Flower pollination	Pollination of flowering plants
Ray optimization	Refraction of light

Table 2.1: Metaheuristic algorithms and inspirations.

tabu search by Glover [27]. Literature review on metaheuristic optimization [87], suggested that it was Fred Glover who coined the word metaheuristics [28].

Each metaheuristic algorithm can have different inspiration from the nature and special rules according to the process of the natural systems. Detailed information about several metaheuristic algorithms can be found in the literature [87, 79]. Inspiration and pioneer papers of several metaheuristic algorithms are given in Table (2.1).

2.4 Metheuristic Methods for ODEs

In this section we present a review of some existing papers trying to solve ODEs with MAs which have some advantages regarding set-up of the problem, storing requirements, interpolation properties, etc. Table (2.2) presents a summary of this papers. It shows the year, the main equations solved, how the candidate solutions are built or expressed, the optimization algorithm and whether if a local search is used or not.

2.5 Conclusion

Metaheuristics can be an efficient way to produce acceptable solutions by trial and error to a complex problem in a reasonably practical time. The complexity of the problem of

Year	Equation	Solution expression	MA	Local Search	Paper
1998	ODEs and PDEs	ANN	BFGS	No	[49]
2005	ODEs	Gram-Schmidt basis functions	GP	No	[46]
2006	ODEs and PDEs	Symbolic expressions	GE	No	[82]
2009	ODEs and PDEs	ANN	GE	BFGS	[83]
2009	1^{st} order diff Eq	ANN	PSO	No	[43]
2010	Simple linear ODEs	Symbolic expressions	CGP	No	[72]
2013	ODEs	Fourier series	PSO	No	[5]

Table 2.2: Some papers about solving ODEs with MAs.

interest makes it impossible to search every possible solution or combination, the aim is to find good feasible solution in an acceptable time scale. There is no guarantee that the best solutions can be found, and we even do not know whether an algorithm will work and why if it does work. The idea is to have an efficient and practical algorithm that will work most of the time and is able to produce good quality solutions. Among the found quality solutions, it can be expected that some of them are nearly optimal, though there is no guarantee for such optimality. But for solving the real world problems, we often have to use both modeling and optimization because modeling makes the objective functions are evaluated using the correct mathematical/numerical model of the problem of interest, while optimization can achieve the optimal settings of design parameters.

Part II

The Main Results

Chapter 3

Fractional Lévy Flight Bat Algorithm (FLFBA)

3.1 Introduction

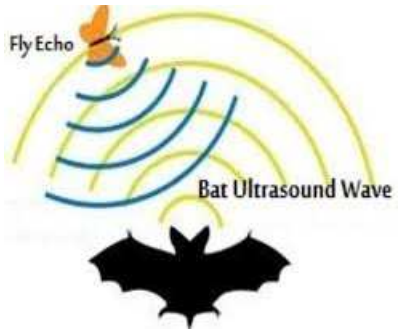
Bat Algorithm (BA) is considered as one of the most well-known metaheuristic algorithms used in optimization problems, which consists of an iterative learning process inspired by bats echolocation behavior in search of preys. More especially, it consists of a number of bats that collectively move on the search space in search of the global optimum. In this thesis we propose the Fractional Lévy Flight Bat Algorithm (FLFBA), which is an improvement of the classical BA algorithm, based on velocity update through fractional calculus and a local search procedure based on an Lévy distribution random walk to enhance the ability of the algorithm to escape from local optimum. The FLFBA is tested using several well-known benchmark functions and the convergence of the algorithm is compared to other recent algorithms [6].

3.2 Related works

3.2.1 Basic bat algorithm

The bat-inspired algorithm, which mimics the echolocation navigation system in detecting and pursuing their preys, was first proposed by Xin-She Yang in 2010 [88]. By emitting

Figure 3.1: Echolocation behavior of bats.



loud sound pulses, the echoes that bounce back from different surrounding objects help bats identify not only their size but also their exact distances when flying in darkness. Bats emit from 10 to 20 ultrasonic sound burst a second with constant frequency (25 KHz to 150 KHz) but as they get closer to their preys they are increased to up to 200 pulses per second. Emitted pulses are as loud as 110 dB but as they get closer to their preys they become quieter. The algorithm is based on the following three idealized rules [90]:

1. Bats use echolocation to measure distance as well as differentiate between food/prey and background barriers (Figure 3.1);

2. Bats fly randomly with velocity v_i from position x_i using a frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. Based on their proximity to target, bats can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$;

3. Although the loudness can vary in many ways, it is assumed that loudness varies from a large pre-defined (positive) value A_0 to a minimum constant value A_{min}.

Initiation

An initial randomly distributed population of N virtual bats is generated with initial positions produced within a preset D-dimensional search space as follows:

$$x_{ij}^0 = x_{min} + (x_{max} - x_{min}) * rand. \tag{3.1}$$

where $i \in [1, \cdots, N]$, $j \in [1, \cdots, D]$ and $rand$ is a random vector with uniformly distributed elements generated in the range $[0, 1]$. The vectors x_{max} and x_{min} contain the upper and lower boundaries in each dimension j, respectively. The initial velocities v_i^0 are generally set to zero.

Generation of New Solutions

Bats navigate by selecting new directions to optimal solutions through the combination of their own and other bats best experience. At each iteration t, a new solution x_i^{t+1} and velocities v_i^{t+1} are updated as follows:

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

$$v_i^{t+1} = v_i^t + (x_i^t - x^g)f_i, \tag{3.3}$$

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

Where $\beta \in [0, 1]$ is uniformly distributed random vector and $f_{min},\ f_{max}$ are the minimum/ maximum frequency of emitted pulse by the bats. The value x^g represents the best global location found so far which is obtained by comparing all the solutions of all N bats at iteration t.

Local Search

After new solutions are generated, a random walk based local search is invoked by a ith bat on the condition that its pulse emission rate r_i is smaller than a random number. The old position x_{old} is modified to obtain a new position x_{new} by,

$$if\ (rand > r_i)\ then,\ x_{new} = x_{old} + \epsilon \bar{A}^t, \tag{3.5}$$

where $\epsilon \in [-1, 1]$ is a random number and $\bar{A}^t = \langle A_i^t \rangle$ is the average loudness of all bats at time steps t.

Solutions and parameters update

As indicated in [88], BA can be considered as a balanced combination of a classical PSO and the intensive local search controlled by both loudness and pulse emission rate. As the bat approaches the prey, it decreases its loudness while increasing the rate of pulse emission. For simplicity, the algorithm starts with an initial set loudness A_0 which is reduced at each iteration until it reaches a A_{min} near zero which represents a bat catching its prey. The bat is guided toward an optimal solution on the basis of the following two design equations,

$$If(rand < A_i^t) \; and \; f(x_i^{t+1}) < f(x_i^t)$$

.

$$A_i^{t+1} = \alpha A_i^t, \tag{3.6}$$

$$r_i^{t+1} = r_i^0 \left[1 - \exp(-\gamma t)\right], \tag{3.7}$$

Where α and γ are constants. The value α, which is similar to a cooling factor of the simulated annealing cooling schedule [45], lies between 0 and 1 while γ is greater than 0. The value A_i^{t+1} represent an updated value of the loudness A_i^t of bat i at time step t. As the time step t tends toward infinity, the rate of pulse emission converges to the initial rate of pulse emission r_i^0 whereas the average loudness of the bat approaches zero expressed as follows,

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \quad as \; t \to \infty. \tag{3.8}$$

3.2.2 Fractional-order calculus

Definition 3.1 *The Riemann-Liouville fractional derivative of an order $\alpha > 0$ of a continuous function $x : (0, +\infty) \to \mathbb{R}$ is given by:*

$${}^{RL}D_{0^+}^{\alpha} x(t) = \frac{1}{\Gamma(m-\alpha)} \frac{d^m}{dt^m} \int_0^t \frac{x(s)}{(t-s)^{\alpha-m+1}} ds, \tag{3.9}$$

where $\Gamma(x)$ is the gamma function, $m - 1 \leq \alpha < m$, and the right-hand side is point-wise defined on $(0, +\infty)$.

Definition 3.2 *Starting with the assumption that a function $x(s)$ satisfies some smooth condition for a finite interval $(0,t)$, the Grüwald-Letnikov fractional derivative definition, which is based on finite difference, with respect to a fractional coefficient $\alpha \in \mathbb{R}$ in an equidistant grid in $[0,t]$ such that:*

$$0 = s_0 < s_1 < \cdots < s_i = (i+1)h < \cdots < s_{n+1} = t = (n+1)h,$$

is given by:

$${}^{RL}D^{\alpha}_{0^+}x(t) = \lim_{h\to 0}\frac{1}{h^{\alpha}}\Delta^{\alpha}_h x(t) = \frac{1}{h^{\alpha}}\Delta^{\alpha}_h x(t) + O(h), \tag{3.10}$$

where

$$\frac{1}{h^{\alpha}}\Delta^{\alpha}_h x(t) = \frac{1}{h^{\alpha}}\left(x(s_{n+1}) + \sum_{i=1}^{n+1}(-1)^i \begin{pmatrix} \alpha \\ i \end{pmatrix} x(s_{n+1-i}) \right), \tag{3.11}$$

$$D^{\alpha}x(t) = \lim_{h\to 0}\left[\frac{1}{h^{\alpha}} \sum_{k=0}^{+\infty} \frac{(-1)^k \Gamma(\alpha+1)x(t-kh)}{\Gamma(k+1)\Gamma(\alpha-k+1)} \right]. \tag{3.12}$$

3.2.3 Bat algorithm modified equations

Following the same principles and steps as in [19], the bat algorithm is described both in discrete and continuous form. In discrete form, the general bat algorithm is given by:

$$\begin{aligned} v[t+1] &= v[t] + (x[t] - x^g[t])f, \\ x[t+1] &= x[t] + v[t+1], \end{aligned} \tag{3.13}$$

and in continuous form by:

$$\begin{aligned} v'[t] &= v[t] + (x[t] - x^g[t])f, \\ x'[t] &= v[t]. \end{aligned} \tag{3.14}$$

The derivatives in Eq.(3.14) can be rewritten as fractional derivatives in the following form:

$$
\begin{aligned}
{}^{GL}D^{\alpha}_{0^+}v[t] &= v[t] + (x[t] - x^g[t])f, \\
{}^{GL}D^{\alpha}_{0^+}x[t] &= v[t].
\end{aligned}
\tag{3.15}
$$

As a result, Eq.(3.15) is rewritten in a form which allows simple numerical computations:

$$
\begin{aligned}
v[t^{n+1}] &= h^{\alpha}\left\{v[t^n] + (x[t^n] - x^g[t^n])f\right\} - \sum_{k=1}^{n+1} s_k v[t_{n+1} - kh], \\
x[t^{n+1}] &= h^{\alpha}v[t^{n+1}] - \sum_{k=1}^{n+1} s_k v[t_{n+1} - kh],
\end{aligned}
\tag{3.16}
$$

where the coefficients s_k are computed in a recursive scheme as follows:

$$
\begin{aligned}
s_0 &= 1, \\
s_k &= \left(1 - \frac{\alpha + 1}{k}\right) s_{k-1},\ k > 0,
\end{aligned}
\tag{3.17}
$$

and $0 = t^0 < \cdots < t^i = ih < \cdots < t^{n+1} = (n+1)h = T$.

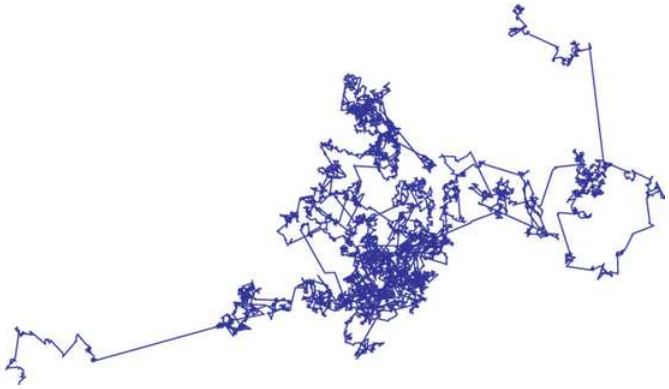
3.2.4 Lévy flight

It is shown in previous studies that different animals and insects follow an Lévy flight behavior in their search for preys or when flying in swarms. The process is defined as a non-Gaussian stochastic random walk kind in which the random step lengths are based on an Lévy distribution (Figure 3.2). For the BA a new location x'_i corresponding to a i^{th} bat is derived by combining a Lévy flight to its old position x_i as follows:

$$
x'_i = x_i + \upsilon \oplus Levy(\lambda). \tag{3.18}
$$

Where υ is a random step size, λ is an Lévy flight distribution parameter and $\oplus$ indicates an entry-wise multiplication. The step size ζ is computed using the Mantegna [54]

Figure 3.2: Example of an Lévy flight trajectory with $\alpha = 1.5$.



algorithm such that,

$$\zeta = \upsilon \oplus Levy(\lambda) \sim 0.01 \frac{u}{|v|^{1/\beta}}(x_i - x_{opt}). \tag{3.19}$$

where u and v are obtained from:

$$\begin{aligned} u \sim N(0, \sigma_u^2),\ \sigma_u = \left(\frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\beta\Gamma[(1+\beta)/2]2^{(\beta-1)/2}} \right)^{1/\beta} \\ , v \sim N(0, \sigma_v^2),\ \sigma_v = 1, \end{aligned} \tag{3.20}$$

and Γ is the Gamma function defined as:

$$\Gamma(y) = \int z^{y-1} e^{-y} dt. \tag{3.21}$$

3.2.5 DE-based location update formula

In order to overcome the entrapment of the BA on local optimum due to a location update equation which is based only on a global best solution. In [91], it is proposed an improved version based on DE. They provide a formula which allows a better search ability at both local and global levels through the following equation:

$$\begin{aligned} v_i^{t+1} &= \omega^t v_i^t + f_i\zeta_1(x_i^t - x^g) + f_i\zeta_2(x_i^t - x_q^t), \\ \zeta_1 &+ \zeta_2 = 1, \\ \zeta_1 &= 1 + (\zeta_{init} - 1)\frac{(T-t)^n}{T^n}, \end{aligned} \tag{3.22}$$

where x_q^t is a randomly selected solution from the population and ζ_1 and ζ_2 are learning factors in the range $[0, 1]$. The ζ_1 is computed based on a relation between an initial ζ_{init}, the maximum number of iterations T, the actual iteration number t, and a nonlinear index n. This modification allows an exploitation in one hand by guiding the velocity update toward the global best x^g and, on the other hand, an exploration based on a third term that is based on a random position x_q^t. The balance between exploration and exploitation is governed by the changes in the term ζ_1.

3.3 Fractional Lévy Flight Bat Algorithm

Even though BA has the advantage of simplicity and flexibility, just like any other meta-heuristics method it still lacks the mechanism of escaping local optimum. The idea is to design an algorithm capable of adjusting itself to the fitness functions landscape making it more robust and applicable to any sort of optimization problem. A sophisticated bat algorithm based on fractional calculus, differential evolution and Lévy flight is developed in this section. The algorithm starts by generating a population of N random locations using Eq. 3.1 and assigning values to the initial velocity vector. The objective function is then evaluated at each position of the initial population and the first global best solution is selected. At each generation and for each bat in the population an update process of the positions is divided, with an equal probability of 50%, between two different mechanisms. The first mechanism starts by producing a new location using the difference between two randomly selected best local solutions multiplied by a random value ε drawn from a uniform distribution and then added to the ith local best solution as in the following:

$$\begin{aligned} s &= \xi(x_q^l - x_p^l) \\ x_n &= x_i^l + s. \end{aligned} \tag{3.23}$$

The fitness function is then evaluated at this new solution and then compared with function value at the ith local best solution. If the new location x_n produces a better result than x_i^l then $x_i^l = x_n$. An updated solution x_i^{t+1} is obtained in the neighborhood of the

corresponding best local solution x_i^l through an Lévy flight search as follows,

$$x_i^{t+1} = x_i^l + 0.01\frac{u}{|v|^{1/\beta}}|\bar{x^l} - x_i^t|, \tag{3.24}$$

where $\bar{x^l}$ is the mean value of the vector of local best solutions. This will assign half of the population in a local search around the selected local best solution which should result in a better exploration procedure. As for the second half of the population, a combination of ED and fractional calculus velocity update equation is utilized. The velocity term is computed based on a fractional differential formula where:

$$\nu_i^t = \sum_{k=0}^{o} s_k v_i^{t-k}, \tag{3.25}$$

where s_k is obtained using Eq.(3.17) and o is the order of the fractional derivative which is selected randomly between the integers 1 and 10. The above process tries to mimics to a certain degree a continuous time random walk where for each individual of the population a different learning period is selected based on the order o of Eq.(3.25). The velocity term computed in Eq.(3.25) is combined with an exploration and exploitation term, through a DE approach, to obtain the updated velocity vector corresponding to each individual of the population by using the following formula:

$$v_i^{t+1} = \omega^t \nu_i^t + f_i\zeta_1(x^g - x_i^t) + f_i\zeta_2(x_i^l - x_q^t), \tag{3.26}$$

where ω^t is an inertia weight, f_i is the frequency, x^g is the global best location, x_i^l is the corresponding local best solution, and x_q^t is a randomly selected solution from the population such that $q \neq i$. The new proposed formula contains several important factors, such as information from previous generations and a combination of exploitation around the global best and an exploration with respect to local best, which should result in a better and consistent convergence of the algorithm. The new computed velocity vector is then added to the actual corresponding location vector as in Eq.(3.4). Once the second half of the population locations are updated, a local search routine, based on the one proposed by [55], is applied to solutions which pulse emission r_i is below a randomly

generated value in the interval $[0, 1]$. The proposed local search is combined with an Lévy flight pattern around the global best solution as indicated below:

$$rA = 0.01\frac{u}{|v|^{1/\beta}}|A_i - \bar{A}| \tag{3.27}$$
$$x_i^{t+1} = x^g(1 + rA),$$

where A_i is the i^{th} related loudness value and $\bar{A}$ is the mean value of the loudness vector. The local best bats are updated and if $\exists x_i^l,\ F(x_i^l) < F(x^g),\ i = 1, \cdots, n$ and the associated loudness value A_i is greater than a random number $\varepsilon \in [0, 1]$ then $x^g = x_i^l$. This is then followed by a reduction in the corresponding ith loudness A_i and an increase in the pulse emission r_i as presented in Eq.(3.6) and Eq.(3.7). The pseudo-code of the modified bat algorithm is presented in Algorithm (1).

3.4 Experimental Results

3.4.1 Parameters settings

The parameters settings of the CS, FDPSO, NBA, ACO, MFO, SFLA and FLFBA are provided in Table (3.1). The maximum number of iterations was set to 50 times the dimension such as for $D = 10$ it is 500, for $D = 20$ it is 1000 and finally for $D = 40$ it is 2000. The search space in all algorithms is restricted to the interval $[-5,\ 5]^D$ since the majority of the benchmark functions has the global optimum solution inside the interval $[-4,\ 4]^D$ or drawn uniformly from this compact. For a better analysis of the results, each optimization procedure was repeated 50 times overall the functions in the three dimensions. It should be noted that the inconsistent results obtained for F_5 in all dimensions were omitted due to a probable bug in the downloaded benchmark source code. The proposed algorithm is evaluated for performance using 24 CEC2015 benchmark functions [68] (see Table (B.1).

Algorithm 1 Pseudo code of FLFBA

N: the number of individuals (bats) in a single population
$MIter$: the maximum number of iterations
G: the frequency of renewing a section of the population
Pa: The portion of the population to be renewed
α, γ, f_{max}, f_{min}, A_0, r_0: classical BA parameters
$t = 0$; Initialize the population using Eq.(3.1)
Initialize velocity vector and local best solutions
Compute fitness and select best solution
while $t < MIter$ **do**
  $f^t = r(f_{max} - f_{min}) + f_{min}$
  Update ω^t, ξ_1^t and ξ_2^t
  for $j = 1, \ldots, n$ **do**
    if $\varepsilon < 0.5$ **then**
      Select two random best local bats
      Generate a new bat x_n using Eq.(3.23)
      if $F(x_n) < F(x_j^l)$ **then**
        $x_j^l = x_n$
      end if
      $x_j^t = x_j^l + \lambda|\bar{x^l} - x_j^{t-1}|$
    else
      Compute v_i^t ▷ Using Eq.(3.25), Eq.(3.22)
      $x_j^t = x_j^{t-1} + v_j^t$
    end if
    if $\varepsilon > r_j$ **then**
      Do a local search using Eq.(3.27)
    end if
    Evaluate $F(x_j^t)$
    Update local best solution
    if $F(x_j^l) < F(x^g)$ and $\varepsilon < A_i$ **then**
      $x^g = x_j^l$
      Update A_j and r_j using Eq.(3.6) and Eq.(3.7)
    end if
  end for
  $t = t + 1$
end while

Table 3.1: Parameters settings

Parameters	**CS**	**FDPSO**	**NBA**	**ACO**	**MFO**	**SFLA**	**FLFBA**
Population size N	30	30	30	30	30	30	30
Loudness A_0	–	–	[1, 2]	–	–	–	[1, 2]
Pulse emission r_0	–	–	[0, 1]	–	–	–	[0, 1]
Frequency $[f_{min}, f_{max}]$	–	–	[0, 1.5]	–	–	–	[0, 2]
ρ, γ	–	–	0.99, 0.9	–	–	–	0.99, 0.9
Probability habitat selection P	–	–	[0.5, 0.9]	–	–	–	–
Compensation rates Doppler echoes C	–	–	[0.1, 0.9]	–	–	–	–
Contraction-expansion coefficient θ	–	–	[0.5, 1]	–	–	–	–
Inertia weight ω	–	0.9	[0.4, 0.9]	–	–	–	[0.2, 0.9]
fractional coefficient α	–	0.632	–	–	–	–	0.632
Cognitive and social components	–	1.5, 1.5	–	–	–	–	–
Search counter / Max Iterations	–	15	–	–	–	5	–
Number of swarms [min, n, max]	–	[1, 2, 3]	–	–	–	–	–
Discovery rate / Step-size	0.25	–	–	2	–	–	–
Deviation-Distance Ratio	–	–	–	1	–	–	–
Intensification Factor	–	–	–	0.5	–	–	–
Memeplex/ Sample Size	–	–	–	40	–	5	–
Offspring Number	–	–	–	–	–	3	–

3.4.2 Results Analysis

In Table (B.2) the average computational time of the selected algorithms using 50 different trials for each benchmark function computed using three variables dimensions, $10-D/20-D/40-D$, is presented. The first observation is the large computational time of both SFLA followed by FDPSO and, for some benchmark functions, CS algorithms which is more than 4 times that of NBA and MFO algorithms. The second observation is the similar performance of ACO and FLFBA time-wise.

To analyze the performance of stochastic algorithms based on computational intelligence we avoid using the parametric tests because the independence, normality, and homoscedasticity assumptions cannot be satisfied, for that reason nonparametric statistical procedures are very practical in this case [22]. In this section we deal with multiple comparisons class of non-parametric analysis and post-hoc procedures to evaluate the performance of FLFBA with respect to remaining five algorithms. Moreover, this analysis helps verify if the proposed FLFBA can significantly improve the accuracy in comparison

with other methods.

The Friedman test aims to distinguish significant differences between two or more algorithms. Its null hypothesis designates sameness of medians between the populations when the alternative hypothesis is given as the reversal of the null hypothesis, its statistic distributed according to the chi-square distribution with $(k-1)$ degrees of freedom. The Friedman Aligned Ranks test is used when the number of compared algorithms is small. The statistical test is evaluated through a chi-square distribution with $(k-1)$ degrees of freedom. The Quade test considered as an alternative test of Friedman by means of the difficulty considerations. In this sense, the rankings computed on each problem could be scaled depending on the differences observed in the algorithm's performances, finding, as a consequence, a weighted ranking analysis of the results sample. It's distributed according to the Fisher distribution with $(k-1)$ and $(k-1)(n-1)$ degrees of freedom where k is the number of the tested algorithms and n is the number of problems considered.

Table (B.3) provides the average rankings of the algorithms achieved by the Friedman, Friedman Aligned, and Quade tests for $D-10$, $D-20$ and $D-40$. Results indicate that SFLA achieves the best average rank by all three tests in all the dimensions while FLFBA was 5^{th} in both 10 and 20 dimensions and finally NBA scored last. A different order of performance is obtained in the $D-40$ case where FLFBA was 4^{th} and ACO came last.

Our experimental study shows that the Friedman and Friedman Aligned Ranks are both distributed according chi-square distribution with 6 degrees of freedom, while the Quade test is distributed according to F-distribution with 6 and 138 degrees of freedom. The Friedman statistic shows an ($F = 41.5312$, p-value $= 2.2757E-7$) for the $10-D$ cases, an ($F = 52.4285$, p-value $= 1.5810E-9$) for the $20-D$ cases and finally an ($F = 80.2857$, p-value$= 4.7076E-11$) for the $40-D$ cases. Iman and Davenport test indicates for the $D-10$ an ($F = 9.3220$, p-value $= 1.4116E-8$), for $D-20$ ($F = 13.1684$, p-value $= 9.4150E-12$) and for $D-40$ ($F = 28.9820$, p-value $= 2.9043E-22$) proposing the existence of significant differences between the tested algorithms.

Contrast Estimation based on medians used to estimate the difference between the presentations of two algorithms in view of all pairwise comparisons. It is particularly helpful to estimate by how far an algorithm outperforms another one. The importance of this

test can be summarized as the estimation of dissimilarity between medians of samples of results. It is important to note that this test cannot give a probability of error related to the refusal of the null hypothesis of equality. In our experimental study, we can calculate the set of estimators of medians directly from the average error results.

Table (B.4) shows the estimations computed for each algorithm in the $D-10$, $D-20$ and $D-40$ cases respectively. Focusing our attention in the rows of the tables, we can underline the performance of FSLA as the best performing algorithm because it have the max number of negative related estimators (attain very low error rates considering median estimators) followed by CS algorithm while FLFBA was 4^{th} in both the $D-20/D-40$ cases.

3.4.3 Post-hoc Procedures

Since the Friedman, Iman-Davenport, Friedman Aligned, and Quade tests can just detect significant differences over the complete multiple comparisons, which makes it incapable to create accurate comparisons between some of the considered algorithms then we can progress with a post-hoc procedure that permit us to establish which algorithms are significantly better/worse.

Tables (B.5) (B.6) and (B.7) show the Holm/ Hochberg/ Hommel, Holland, Rom, Finner and Li procedures for all six algorithms in the $D-10$, $D-20$ and $D-40$ cases respectively for $alpha = 0.05$.

In order to better show the differences between the three tests and their respective approximations for obtaining the p-value (also named unadjusted p-values), of every hypothesis, we will compute the unadjusted p-values for the selected algorithms. Numerous dissimilarities can be clarified; Friedman test shows a lower power than the Friedman Aligned test (the unadjusted p-values are considerably lower). Within a multiple comparison tests the p-values are not appropriate because it does not consider the remaining comparisons going to the family, it just represents the probability error of a certain comparison.

Adjusted p-values can treat this problem. They are suitable to be employed since they offer more information in a statistical analysis. They assume the accumulated family error, also they can be evaluated directly with every selected significance level α. Therefore,

Tables (B.8), (B.9) and (B.10)) show the p-values obtained, using the ranks computed by the Friedman, Friedman Aligned, and Quade tests, respectively for the 8 considered post hoc procedures for the $D-10$, $D-20$ and $D-40$ cases respectively.

The Friedman test illustrates a significant performance of FSLA over the remaining algorithms while the Friedman Aligned test validate its improvement for each post-hoc procedure considered except Bonferroni-Dunn which fails to emphasize the significant differences between them. It should be noted that NBA and MFO are interchangeably omitted from the results due to their worst scores. The Finner and Li tests have the lowest p-values in the comparisons displaying the most powerful behavior. Finally, the Quade test also confirms the order of the three first performing algorithms, i.e. FSLA, CS and FDPSO, and indicates different positions between FLFBA and ACO. This result support the conclusion that FSLA performed better than the remaining algorithms while FLFBA had an intermediate position in the scores tables.

3.5 Conclusion

This chapter introduces a hybrid version of BA that is called FLFBA. The proposed algorithm is based on FC and LF techniques with DE strategies for solving optimization problems. FLFBA has been validated using several benchmarks functions and compared to five algorithms that are CS, FDPSO, SFLA, ACO, MFO and NBA. Several nonparametric statistical tests using an average of the difference between the computed optimal fitness function value and the true global optimum function value were conducted in order to analyze the performance of the FLFBA algorithm.

FLFBA showed a distinguished performance in comparison to MFO and NBA but failed to provide similar or better results than FSLA, CS, and FDPSO. Also studies on the time taken to perform the iterations for each algorithm indicate that the FLFBA was in most of the cases much faster than FSLA, CS and FDPSO, but slower than NBA and MFO.

As shown with our experiments FLFBA uses a balanced combination of the advantages of the successful proprieties of FC, LF and DE which provides a superior performance than the NBA algorithms in terms of accuracy and efficiency.

Chapter 4

Application of FLFBA in Optimizing IVP

4.1 Introduction

To illustrate the FLFBA's performance and to demonstrate its computationally efficiency, we select - as a studied problem - the bacterial population growth models that are the logistic growth and the exponential growth models by taking a uniform step size h.

The main motivation in the selection of the application examples comes from the great importance of the exponential equation in modeling any phenomena where a quantity is allowed to undergo unrestrained growth, while the logistic differential equations [81] are an ODE whose solution is a logistic function, they are useful in various other fields as well, as they often provide significantly more practical models than exponential ones which fail to take into account constraints that prevent indefinite growth, and logistic functions correct this error. They are also useful in a variety of other contexts, including machine learning, chess ratings, cancer treatment (i.e. modeling tumor growth), economics, and even in studying language adoption. The logistic function is shown to be the solution of the Riccati equation, some second-order nonlinear ODEs and many third-order nonlinear ODEs [47].

In this chapter, the IVP is formulated as an optimization problem [61, 62, 63, 64, 65] it will be solved with FLFBA compared to several methods including Euler's methods (Explicit

Euler, Midpoint method and Backward Eulers), Runge–Kutta methods (RK4, Heuns (RK2)) and predictor–corrector methods (Adams–Bashforth–Moulton method (ABM)). Then FLFBA is compared with three MAs that are: Artificial Bee Colony Algorithm (ABCA) inspired by the behavior of honey bees [59, 40], Bat Algorithm (BA) simulates the echolocation behavior of bats [88] and Flower Pollination Algorithm (FPA) inspired by the flower pollination process of flowering plants [90] to examine which algorithm find the best numerical solutions with the best effectiveness for the studied problem. All computations were performed on an MSWindow 2007 professional operating system in the Matlab environment version R2013a compiler on Intel Duo Core 2.20 GHz. PC.

4.2 Problem Formulation

We consider the general Cauchy problem as in Eq. (1.1):

$$\begin{cases} \acute{y} = f(x, y) \\ y(t_0) = y_0 \end{cases},$$

where t is the independent variable and $y = y(t)$ is the dependent variable. By using the classical assumption:

$$f : [t_0 - T, t_0 + T] \times [y_0 - Y, y_0 + Y] \to \mathbb{R},$$

is continuous and satisfies the Lipschitz condition:

$$|f(t, y_1) - f(t, y_2)| \leq L\,|y_1 - y_2|\,,$$

it results there exists a single solution y. There are many methods used to find the solution, but, in practice, we always solve the problem by using numerical methods, like Runge-Kutta or Euler methods but these classical mathematical tools are not very precise. The main goal of this thesis is to underline the possibility of using a different method, based on metaheuristic algorithms like FLFBA.

4.2.1 Objective Function

Finding the values of the unknown function $y = y(t), y : [a, b] \to \mathbb{R}$, according to a finite set of equidistant values of the independent variable $t_0 = a < t_1 < ... < t_n = b, t_i = a + ih, h = \dfrac{b-a}{n}$. We denote by $y_i = y(t_i), i = 1...n$ the values of the unknown function y, in accordance with the given division. Thus, the vector $(y_1, y_2, ..., y_n)$ is an admissible solution. We will consider the population as being a subset of admissible solutions. Given an instant t, we denote the population by $Y(t)$. One individual $y = (y_1, y_2, ..., y_n)$ is characterized by the values y_i. The individuals in a natural population are, more or less, adapted. Thus, in order to simulate natural selection, we will select, in each stage, only one subset of individuals, namely those who are best adapted. The surplus of individuals is eliminated, taking into account the decreasing values of the objective function. In order to evaluate each individual, we will use the following approximate formula (finite difference formula) for the derivative:

$$\acute{y}(t_i) \approx \frac{y_i - y_{i-1}}{h}, \qquad \left| \acute{y}(t_i) - \frac{y_i - y_{i-1}}{h} \right| \leq const.h.$$

Consequently, the discrete form of the Cauchy problem will be:

$$\frac{y_i - y_{i-1}}{h} = f(t_i, y_i), \qquad i = 1...n. \tag{4.1}$$

The above system is, generally, nonlinear. Finding the vector $(y_1, y_2, ..., y_n)$ which satisfies the above conditions is our goal. Of course, for an admissible solution, we do not have the equality in Eq. (4.1) and, consequently, we have to consider the error formula:

$$\left(\frac{y_i - y_{i-1}}{h} - f(t_i, y_i) \right)^2.$$

The objective function associated to an individual $y = (y_1, y_2, ..., y_n)$ will be:

$$F(y) = \sum_{i=1}^{n} \left(\frac{y_i - y_{i-1}}{h} - f(t_i, y_i) \right)^2. \tag{4.2}$$

An individual from $Y(t)$ will be better adapted if it implies a smaller value of the function F. Each individual may suffer some modifications, which may be hazardous, we will consider that $y_i \pm \varepsilon$ is a mutation for y_i.

4.2.2 Consistency

We denote by u_t the best adapted individual in the population, at the instance t, i.e. the individual in the population which has the minimum value of the function F. In [53] it's already stated that the sequence $(u_t)_{t\geq 0}$ converges, its limit being the solution of the optimization problem $\inf F$. While the solution is the limit of a convergent sequence, by applying the optimization algorithms, the following assertion is true:
For $\varepsilon > 0$, there is a $(y_1, y_2, ..., y_n)$ such that:

$$F(y) = \sum_{i=1}^{n} \left(\frac{y_i - y_{i-1}}{h} - f(t_i, y_i) \right)^2 < \varepsilon,$$

it results there is a $y = (y_1, y_2, ..., y_n)$ such that:

$$\left| \frac{y_i - y_{i-1}}{h} - f(t_i, y_i) \right| < h,$$

taking into account the approximation of the derivative, we have:

$$|\acute{y}(t_i) - f(t_i, y_i)| \leq \left| \acute{y}(t_i) - \frac{y_i - y_{i-1}}{h} \right| + \left| \frac{y_i - y_{i-1}}{h} - f(t_i, y_i) \right| < Ch,$$

when C denotes a positive constant. The last relation shows that the final value $y = (y_1, y_2, ..., y_n)$ is an approximate solution of the Cauchy problem, for small values of h.

4.3 Population Growth Models

In this section we explain briefly the two population growth models:

4.3.1 Exponential growth

Suppose that $P(t)$ describes the quantity of a population at time t. For example, $P(t)$ could be the number of milligrams of bacteria in a particular beaker for a biology experiment at a time t. A model of population growth tells plausible rules for how such a population changes over time. The simplest model of population growth is the exponential model, which assumes that there is a constant parameter r, called the growth parameter, such that:

$$\acute{P}(t) = rP(t),$$

holds for all time t. This differential equation it self might be called the exponential differential equation, because its solution is:

$$P(t) = P_0 e^{rt}, \tag{4.3}$$

where $P_0 = P(0)$ is the initial population. One noticeable feature of the exponential model is that, when r is positive, the population always grows larger and larger without any finite limit. This kind of growth may be a good model for a new population of bacteria in a beaker, but it does not hold for a long time. It is easy to see that the equation would imply a population of bacteria that ultimately outgrew the beaker and even outgrew the planet earth, since the mass of the bacteria would ultimately exceed the mass of the earth. Such a model is therefore absurd to model a system for long periods of time. The fundamental difficulty is that the exponential differential equation ignores the fact that there are limits to resources needed for the population to grow. It ignores the needs for food, oxygen, and space; and it ignores the accumulation of waste products that inevitably arise. The logistic curve gives a much better general formula for population growth over a long period of time than does exponential growth.

4.3.2 Logistic growth

An alternative model was proposed by Verhulst in 1836 [81] to allow for the fact that there are limits to growth in all known biological systems. He proposed what is now called the logistic differential equation. The equation involves two positive parameters. The first

parameter r is again called the growth parameter and plays a role similar to that of r in the exponential differential equation. The second parameter K is called the carrying capacity. The logistic differential equation is written:

$$\acute{P}(t) = rP(t)[\frac{K - P(t)}{K}].$$

Equivalently, in terms of the d notation, the logistic differential equation is:

$$\frac{dP}{dt} = rP[\frac{K - P(t)}{K}].$$

Note that when $P(t)$ is very small, then $P(t)/K$ is close to 0, so the entire factor $[\frac{K-P(t)}{K}]$ is close to 1 and the equation itself is then close to $\acute{P}(t) = rP(t)$; we then expect that the population grows approximately at an exponential rate when the population is small. On the other hand, if $P(t)$ gets to be near K, then $P(t)/K$ will be approximately 1, so $[\frac{K-P(t)}{K}]$ will be approximately 0, and the logistic differential equation will then say approximately $\acute{P}(t) = rP(t)0 = 0$. The growth rate will be essentially 0, so the population will not grow significantly more. The solution of the logistic differential equation is:

$$P(t) = \frac{P_0 K}{P_0 + (K - P_0)e^{-rt}}, \tag{4.4}$$

where $P_0 = P(0)$ is the initial population. This formula is the logistic formula. It tells the equation for the logistic curve.

4.4 Numerical Experiments

The implementation of any numerical method could turn difficult because it is necessary to take into account several issues as the discretization order, the algorithm stability, the convergence speed, how to fulfill the boundary conditions, etc. In the methods described in this thesis, the original problem is transformed into an optimization one according to Eq. (4.2). For making a quantitative comparison, this section is devoted to compare the FLFBA with other algorithms, such as other metaheuristic approaches or more traditional numerical methods, two ordinary differential equations (linear and nonlinear IVP)

have been solved with traditional numerical methods (see Table 4.3) and metaheuristic algorithms (see Table 4.6).

4.4.1 Application example

Consider a bacterial population growth problem, when the initial population is 3 *milligrams* (mg) of bacteria, the carrying capacity is $K = 100\ mg$, and the growth parameter is $r = 0.2\ hour^{-1}$. We want to find the solutions of the differential equations satisfied by this population by means of FLFBA, ABCA, BA, FPA and more traditional numerical methods and comparing between their performances.

Problem 4.1 ***Exponential growth model*** *The exponential growth model is considered as a linear first order IVP, hence based on Eq. (4.3) the exponential differential equation is given by*

$$P(t) = 3e^{0.2t}.$$

Problem 4.2 ***Logistic growth model*** *The logistic differential equation related to our example is considered as a Bernoulli differential equation (and also a separable nonlinear first order IVP), solving it using either approach gives the solution as in Eq. (4.4)*

$$P(t) = \frac{(3)(100)}{3 + (100 - 3)e^{-0.2t}} = \frac{300}{3 + 97e^{-0.2t}}$$

.

4.4.2 Parameters adopted to solve IVP

FLFBA, ABCA, BA and FPA are an optimization instrument. Then, the essential differential equation is converting into discretization form Eq. (4.2). The difference formula is used to convert differential equation into discretizations form when the derivative term is replaced in the discretized form by a difference quotient for approximations. The IVP related parameters are as follows:

1. The number of interior nodes ($n = 9$).

Parameters	Value
Dimension of the search variables (dim)	10
Maximal number of generations (iterations) (M)	100
Population size (pop)	30
The maximal and minimal pulse rate $(r0Max, r0Min)$	$(1, 0)$
The maximal and minimal frequency $(freqDMax, freqDMin)$	$(2, 0)$
The maximal and minimal loudness $(AMax, AMin)$	$(2, 0)$
gamma	0.9
alpha	0.99
The maximal and minimal inertia weight $(wMax, wMin)$	$(0.9, 0.2)$

Table 4.1: Parameters adopted to generate FLFBA.

2. The initial condition in our examples is considered by 3 $milligrams$ (mg) of bacteria and the interval between which the differential equation is $t \in [0, 50]$.

3. The interval of the IVP is equally partitioned into $(n + 1)$ equidistant subintervals with the length $h = (b - a)/n + 1$. Since $t \in [0, 50]$ in our example, hence the step size $h = 5$.

4. The number of generations is set to 100 and the population size is set to 30 for all MAs used in this study.

5. For a better analysis of the results, a Monte Carlo simulation is performed (i.e. we run the program several times for the same testing problem) so each optimization procedure was repeated 50 times for all MAs and in all dimensions.

6. The objective function:

$$F\left(y_1, y_2, ..., y_{10}\right) = \sum_{j=1}^{10} \left(\frac{y_j - y_{j-1}}{h} - f\left(t_{j-1}, y_{j-1}\right)\right)^2 = \sum_{j=1}^{10} \left(\frac{y_j - y_{j-1}}{h} - y_{j-1}\right)^2$$

.

Table (4.1) indicates the different parameters used to generate FLFBA [6]. Table (4.2) gives the parameters adopted to generate BA, FPA and ABCA (for more details about these three algorithms see [88], [90] and [59] respectively).

Parameters	BA	FBA	ABCA
Dimension of the search variables (d)	10	10	10
Number of generations (N)	100	100	100
Population size (n)	30	30	30
Loudness (constant or decreasing) (A)	0.5	/	/
Pulse rate (constant or decreasing) (r)	0.5	/	/
Probabibility switch (p)	/	0.8	/

Table 4.2: Parameters adopted to generate BA, FPA and ABCA.

4.4.3 Comparison of FLFBA with numerical methods

In this subsection, we look into several methods of obtaining the numerical solutions to ordinary differential equations (ODEs) in which all dependent variables (x) depend on a single independent variable (t).

The IVPs will be handled with several methods including Euler's methods (Explicit Euler, Midpoint method and Backward Eulers), Runge–Kutta methods (RK4, Heuns (RK2)) and predictor–corrector methods (Adams–Bashforth–Moulton method(ABM)). In Matlab we plot the numerical results together with the (true) analytical solution. The results are depicted in Figure (4.1) and listed in Table (4.3).

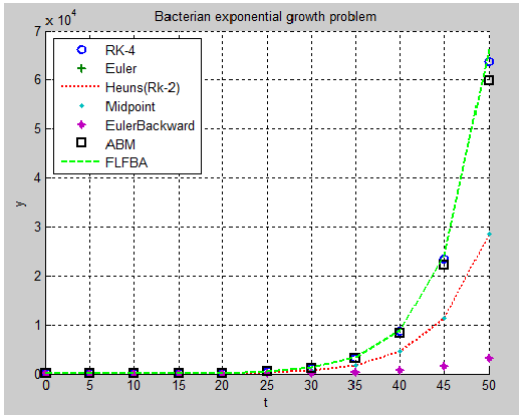
Comparison of exact results with those of numerical methods and FLFBA show that the RK4 method is better than Heun's method and ABM's method, while Euler's method is the worst in terms of accuracy with the same step-size, while the FLFBA approach gives the best solution since it does not depend on the type of differential equation i. e., is based on velocity update through fractional calculus and a local search procedure based on an Lévy distribution random walk. The absolute error of the proposed methods are made in the Table ((4.4)) and summarized via Figure (4.2).

Starting with the Euler method, since it is easy to understand and simple to program. Even though its low accuracy keeps it from being widely used for solving ODEs, it gives us a clue to the basic concept of numerical solution for a differential equation simply and clearly. The error of Heun's method is $O(h^2)$ (proportional to h^2), while the error of Euler's method is $O(h)$. Although Heun's method is a little better than the Euler method, it is still not accurate enough for most real-world problems. The global error of the midpoint method is of order $O(h^2)$. Thus, while more computationally intensive than

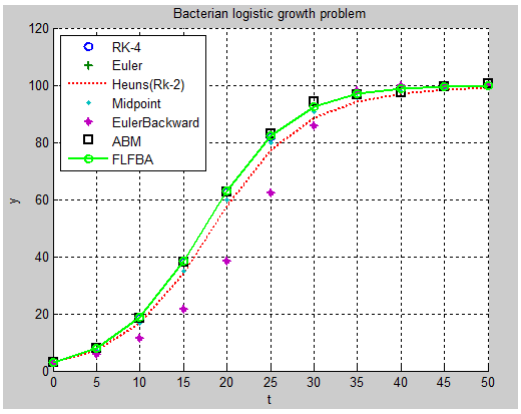
i	x_i	**Exact**	Expl Euler	RK4	Heuns	Midpoint	Back Euler	ABM	**FLFBA**
					Problem 1				
0	0	**3.0000**	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000	**3 0000**
1	5	**8.0000**	6.0000	8.0000	8.0000	8.0000	6.0000	8.1250	**8.0000**
2	10	**22.000**	12.000	22.000	19.000	19.000	12.000	22.005	**22 000**
3	15	**60.000**	24.000	60.000	47.000	47.000	24.000	59.597	**60.000**
4	20	**164.00**	48.000	161.00	117.00	117.00	48.000	160.08	**167.00**
5	25	**445.00**	96.000	437.00	293.00	293.00	96.000	429.57	**453.00**
6	30	**1210.0**	192.00	1184.0	732.00	732.00	192.00	1152.8	**1236.0**
7	35	**3290.0**	384.00	3207.0	1831.0	1831.0	384.00	3093.7	**3319.0**
8	40	**8943.0**	768.00	8684.0	4578.0	4578.0	768.00	8302.7	**8977.0**
9	45	**24309**	1536.0	23520	11444	11444	1536.0	22282	**24346**
10	50	**66079**	3072.0	63700	28610	28610	3072.0	59798	**66120**
					Problem 2				
0	0	**3.00000**	3.00000	3.00000	3.00000	3.00000	3.00000	3.00000	**3.00000**
1	5	**7.75510**	5.91000	7.73340	7.23540	7.25650	5.91000	7.73330	**7.75550**
2	10	**18.6017**	11.4707	18.5208	16.5923	16.7499	11.4707	18.5207	**18.6044**
3	15	**38.3174**	21.6257	38.1583	34.0973	34.8446	21.6257	38.1584	**38.3231**
4	20	**62.8060**	38.5746	62.6348	57.6171	59.6999	38.5746	62.5191	**62.8112**
5	25	**82.1112**	62.2692	81.9715	77.1952	79.9782	62.2692	83.0479	**82.1157**
6	30	**92.5800**	85.7639	92.4564	88.4623	90.5498	85.7639	94.1325	**92.5845**
7	35	**97.1360**	97.9733	97.0456	94.2223	95.4540	97.9733	96.7510	**97.1384**
8	40	**98.9270**	99.9589	98.8735	97.1106	97.7738	99.9589	97.3996	**98.9275**
9	45	**99.6026**	100.000	99.5749	98.5553	98.8988	100.0000	99.4568	**99.6024**
10	50	**99.8534**	100.000	99.8402	99.2776	99.4523	100.0000	100.493	**99.6024**

Table 4.3: Comparison of FLFBA with numerical methods.

Figure 4.1: Solution of bacterial growth problems by numerical methods vs. FLFBA.



(a) Problem 1

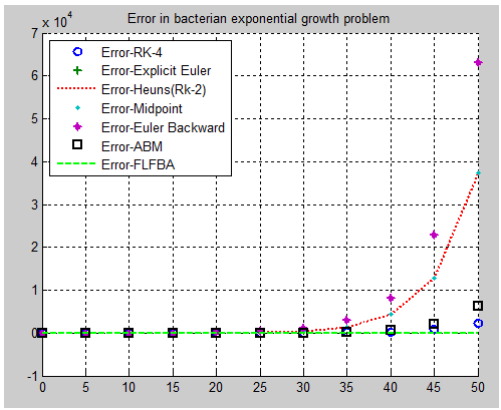


(b) Problem 2

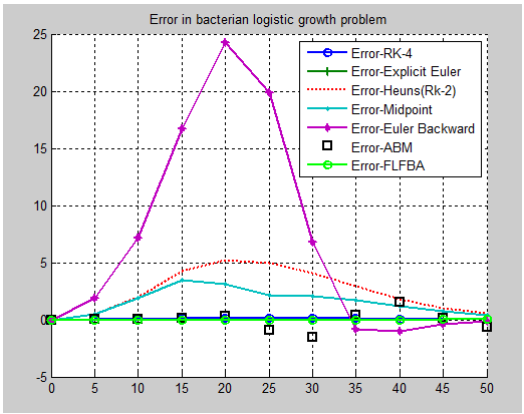
i	x_i	Expl Euler	RK4	Heuns	Midpoint	Back Euler	ABM	**FLFBA**
					Problem 1			
0	0	00.000	00.000	00.000	00.000	00.000	00.000	**00.000**
1	5	02.000	00.000	00.000	00.000	02.000	01.000	**00.000**
2	10	10.000	00.000	03.000	03.000	10.000	00.000	**00.000**
3	15	36.000	00.000	13.000	13.000	36.000	04.000	**00.000**
4	20	116.00	03.000	47.000	47.000	116.00	39.000	**03.000**
5	25	349.00	08.000	152.00	152.00	349.00	154.00	**08.000**
6	30	1018.0	26.000	478.00	478.00	1018.0	572.00	**26.000**
7	35	2906.0	83.000	1459.0	1459.0	2906.0	196.20	**29.000**
8	40	8175.0	259.00	4365.0	4365.0	8175.0	640.30	**34.000**
9	45	22773	789.00	12865	12865	22773	202.71	**37.000**
10	50	63007	2379.0	37469	37469	63007	628.14	**41.000**
					Problem 2			
0	0	00.0000	0.0000	0.0000	0.0000	00.0000	0.0000	**0.0000**
1	5	01.8451	0.0217	0.5197	0.4986	01.8451	0.0218	**0.0004**
2	10	07.1310	0.0809	2.0094	1.8518	07.1310	0.0810	**0.0027**
3	15	16.6917	0.1591	4.2201	3.4728	16.6917	0.1590	**0.0057**
4	20	24.2314	0.1712	5.1889	3.1061	24.2314	0.2869	**0.0052**
5	25	19.8420	0.1397	4.9160	2.1330	19.8420	0.9367	**0.0045**
6	30	06.8161	0.1236	4.1177	2.0302	06.8161	1.5525	**0.0045**
7	35	00.8373	0.0904	2.9137	1.6820	00.8373	0.3850	**0.0024**
8	40	01.0319	0.0535	1.8164	1.1532	01.0319	1.5274	**0.0005**
9	45	00.3974	0.0277	1.0473	0.7038	00.3974	0.1458	**0.0002**
10	50	00.1466	0.0132	0.5758	0.4011	00.1466	0.6396	**0.0002**

Table 4.4: Absolute error between exact solution and different methods.

Figure 4.2: Plot of the error between exact solution and different methods.



(a) Problem 1

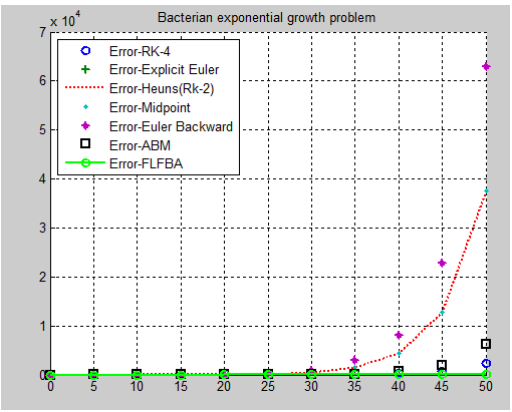


(b) Problem 2

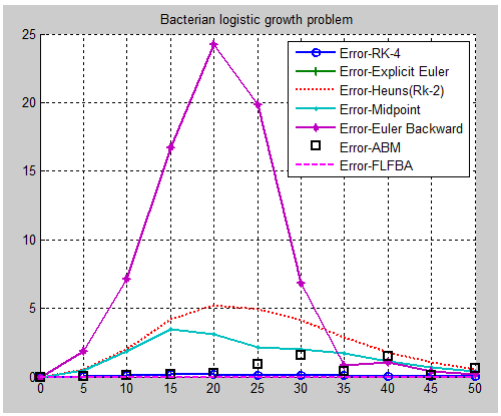
	Expl Euler	RK4	Heuns	Midpoint	Back Eulers	ABM	FLFBA
Problem 1	63019.6056	2391.1697	37481.3761	37481.3761	63019.6056	6294.4	33
Problem 2	24.2366	0.17644	5.1942	3.4785	24.2366	1.5480	0.000

Table 4.5: Maximum error of ode45 vs. differrent numerical methods with step size h= 5.

Figure 4.3: Plot of absolute error between ode45 routine and different methods.



(a) Problem 1



(b) Problem 2

Euler's method, the midpoint method's error generally decreases faster as $h \to 0$. The fourth-order Runge–Kutta (RK4) method having a truncation error of $O(h^4)$ is one of the most widely used methods for solving differential equations The Adams–Bashforth–Moulton (ABM) scheme needs only two function evaluations (calls) per iteration, while having a truncation error of $O(h^5)$.

From Table (4.5) and Figure (4.3) that show the maximum error of MATLAB built-in routine "ode 45" compared with different numerical methods and FLFBA approach with step size $h = 5$, we can see that the predictor–corrector methods such as the ABM method gives a better numerical solution with less error and shorter computation time (see Table (4.8)) than the MATLAB built-in routine "ode45", as well as the FLFBA (but, a general conclusion should not be deduced just from one example).

4.4.4 Comparison of FLFBA with metaheuristic algorithms

In this subsection, the IVP is formulated as an optimization problem (Eq. 4.2) solved with three metaheuristics that are: ABCA inspired by the behavior of honey bees, BA simulates the echolocation behavior of bats and FPA inspired by the flower pollination

i	x_i	Exact	BA	FPA	ABCA	**FLFBA**
				Problem 1		
0	0	03.000	06.000	04.000	11.000	**03.000**
1	5	08.000	15.000	12.000	20.000	**08.000**
2	10	22.000	33.000	29.000	39.000	**22.000**
3	15	60.000	75.000	69.000	83.000	**60.000**
4	20	164.00	183.00	166.00	193.00	**167.00**
5	25	445.00	457.00	452.00	456.00	**453.00**
6	30	1210.0	1238.0	1215.0	1246.0	**1236.0**
7	35	3290.0	3323.0	3319.0	3330.0	**3319.0**
8	40	8943.0	8982.0	8951.0	8988.0	**8977.0**
9	45	24309	24352	24346	24355	**24346**
10	50	66079	61260	66120	66130	**66120**
				Problem 2		
0	0	03.0000	03.0015	03.0010	03.0021	**03.0000**
1	5	07.7551	00.7559	00.7556	00.7561	**07.7555**
2	10	18.6017	18.6063	18.6063	18.6063	**18.6044**
3	15	38.3174	38.3258	38.3249	38.3278	**38.3231**
4	20	62.8060	62.8159	62.8149	62.8179	**62.8112**
5	25	82.1112	82.1228	82.1208	82.1237	**82.1157**
6	30	92.5800	92.5927	92.5909	92.5936	**92.5845**
7	35	97.1360	97.1493	97.1477	97.1501	**97.1384**
8	40	98.9270	98.9412	98.9397	98.9423	**98.9275**
9	45	99.6026	99.6180	99.6160	99.6187	**99.6024**
10	50	99.8534	99.8697	99.8675	99.8708	**99.6024**

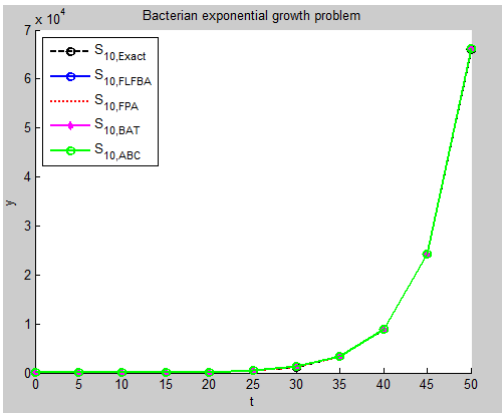
Table 4.6: Comparison of FLFBA with MAs.

process of flowering plants as well as the FLFBA, by focusing on the performance of these three algorithms compared to FLFBA's performance to examine which one finds the best numerical solutions with the best effectiveness for the studied problems. The obtained results, the comparison of the proposed algorithms to the exact solution are shown in Table (4.6) and summarized via Figure (4.4.)

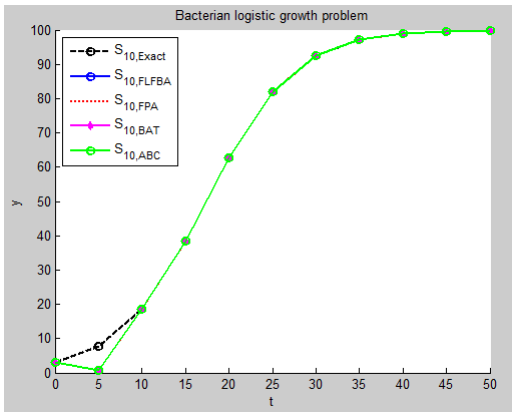
After a comparison between the exact solution and the algorithms outcomes of the chosen examples; the results found that FLFBA is very adequately precise than ABCA, BA and FPA in both exponential and logistic growth models since it possesses the smallest error. The absolute error of the proposed algorithms are made in the Table (4.7) and summarized via Figure (4.5).

The comparison between the performances of BA, FPA, ABCA and FLFBA face to the exact results confirm that FLFBA is better because it has a very close curve to the

Figure 4.4: Solution of bacterial growth problems by ABCA, FBA, BA and FLFBA.



(a) Problem 1

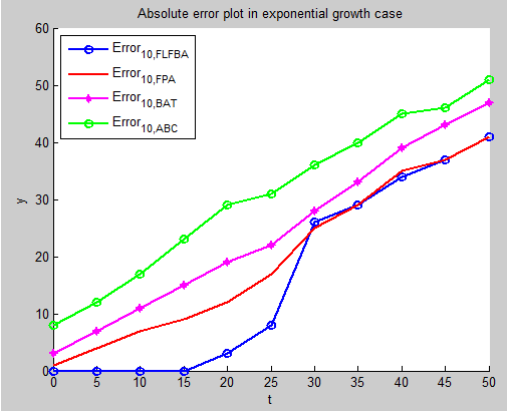


(b) Problem 2

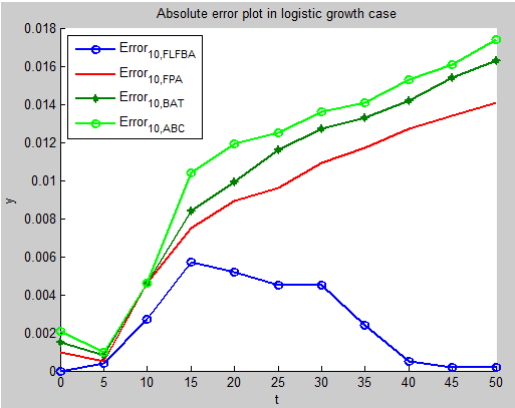
i	x_i	BA	FPA	ABCA	**FLFBA**
			Problem 1		
0	0	03.000	01.000	08.000	**00.000**
1	5	07.000	04.000	12.000	**00.000**
2	10	11.000	07.000	17.000	**00.000**
3	15	15.000	09.000	23.000	**00.000**
4	20	19.000	12.000	29.000	**03.000**
5	25	22.000	17.000	31.000	**08.000**
6	30	28.000	25.000	36.000	**26.000**
7	35	33.000	29.000	40.000	**29.000**
8	40	39.000	35.000	45.000	**34.000**
9	45	43.000	37.000	46.000	**37.000**
10	50	47.000	41.000	51.000	**41.000**
			Problem 2		
0	0	0.0015	0.0010	0.0021	**0.0000**
1	5	0.0008	0.0005	0.0010	**0.0004**
2	10	0.0046	0.0046	0.0046	**0.0027**
3	15	0.0084	0.0075	0.0104	**0.0057**
4	20	0.0099	0.0089	0.0119	**0.0052**
5	25	0.0116	0.0096	0.0125	**0.0045**
6	30	0.0127	0.0109	0.0136	**0.0045**
7	35	0.0133	0.0117	0.0141	**0.0024**
8	40	0.0142	0.0127	0.0153	**0.0005**
9	45	0.0154	0.0134	0.0161	**0.0002**
10	50	0.0163	0.0141	0.0174	**0.0002**

Table 4.7: Absolute error between the exact solution and MAs.

Figure 4.5: Plot of absolute error of bacterial growth problems between the exact solution and MAs.



(a) Problem 1



(b) Problem 2

exact curve contrary to the other methods. In both representations of the absolute error (tabular and graphical), FLFBA method offers a very negligible absolute error compared to the other methods.

4.4.5 Time taken for the algorithms

The major factors to be considered in evaluating/comparing different numerical methods is the accuracy of the numerical solution and its computation time. Table (4.8) shows the time taken for the different studied algorithms. In this comparison, we can say that in some cases the MAs can achieve a more accurate solution using less time consuming than the numerical methods because of in the MAs the solutions obtained are coded in a more compact way requiring significantly less amount of memory.

It is important to note that the evaluation/comparison of numerical methods is not so simple because their performances may depend on the characteristic of the problem at hand. It should also be noted that there are other factors to be considered, such as stability, versatility, proof against runtime errors, and so on.

4.5 Conclusion

Throughout this chapter, application of standard ABCA, BA, FPA, some numerical methods for solving IVP compared to FLFBA is discussed when they are used as a tool for optimize numerically the IVPs arising in environmental field that is differential equations

Algorithm	Problem 1	Problem 2
Expl Euler	41×10^{-4} s	38×10^{-4} s
Rk4	70×10^{-4} s	21×10^{-4} s
Heuns	57×10^{-4} s	23×10^{-4} s
Midpoint	52×10^{-4} s	24×10^{-4} s
Back Eulers	51×10^{-4} s	23×10^{-4} s
ABM	51×10^{-4} s	23×10^{-4} s
FLFBA	32×10^{-4} s	21×10^{-4} s
FPA	34×10^{-4} s	21×10^{-4} s
BA	34×10^{-4} s	22×10^{-4} s
ABCA	35×10^{-4} s	23×10^{-4} s

Table 4.8: Time taken for the algorithms.

describing the growth phenomena of such population in both exponential and logistic cases with an initial population via a chosen example.

In the exponential growth problem, results show a population growing always faster without any bond. In reality this model is unrealistic because environments impose limitations to population growth. A more accurate model postulates that the relative growth rate $\frac{\acute{P}}{P}$ decreases when P approaches the carrying capacity K of the environment.

But in the case of logistic growth problem, results show the logistic curve. Note that it has roughly the shape of an elongated S (and it is in fact sometimes called the $S-shaped$ *curve*). The population initially grows slowly but steadily. Then the growth speeds up and the curve moves more steeply upward. As the population gets closer to the carrying capacity $K = 100$, the growth slows and the curve gets more horizontal again. In fact the population never appears to reach the carrying capacity, but instead seems to approach it as an asymptote.

After a comparison between the exact solutions and the algorithms outcomes; FLFBA was found exponentially better than the other methods by giving accurate solutions with smallest amount error.

General Conclusion

In terms of this research work we can detect that the current trend is to use nature-inspired MA to tackle such difficult problems, and it has been shown that metaheuristics are surprisingly very efficient. For this reason, the literature of metaheuristics has expanded tremendously in the last two decades [90, 79]. Up to now, researchers have only used a very limited characteristics inspired by nature, and there is room for more algorithm development. Optimization is paramount in many applications such as engineering and industrial designs. Obviously, the aims of optimization can be anything (to minimize the energy consumption, to maximize the profit, output, performance and efficiency, etc). There are many reasons for such popularity. From the algorithm analysis point of view, these algorithms tend to be:

1. Flexible, highly adaptable, and yet easy to implement.
2. Algorithmic procedures are quite simple and flexible, and yet efficient in practice.
3. The high efficiency of these algorithms makes it possible to apply them to a wide range of problems in diverse applications.
4. Their multiple agents interact and exchange information, following simple rules, show complex and self-organized behavior.

4.6 Bilan of contributions

The echolocation conduct of bats is the principle motivation behind the idea of creating and presenting a BA hybridization that is FLFBA based in FC and LF techniques with DE strategies for solving optimization problems. FLFBA [6] has been validated using several benchmarks functions distributed via four sets: Set of separable functions, set of

low or moderate conditioning problems, set of high conditioning and unimodal functions, set of multimodal functions with adequate global structure and the set of multimodal functions with weak global structure.

In order to prove the efficacy of the proposed algorithms, a series of experiments was conducted hence the results were compared with a variety of well-known and recent algorithms that are CS, FDPSO, SFLA, ACO, MFO and NBA.

With respect to the outcomes got from every each benchmark function and each test in each studied case. The FLFBA giving significantly improved result on its results as it mentioned in the majority of the cases obviously superior to the thought about algorithms by means of almost used tests. Besides the FLFBA achieved a similar good performance in some cases but failed to give comparable or preferable outcomes over the other algorithms in other few cases.

Regarding to the results obtained after studying the time taken to perform the iterations for each algorithm indicate that the FLFBA was in most of the cases much faster than FSLA, CS and FDPSO, but slower than NBA and MFO.

As shown with our experiments FLFBA uses a balanced combination of the advantages of the successful proprieties of FC, LF and DE while the basic BA employ the advantages inspired by the fantastic behavior of echolocation of bats which make the FLFBA much superior to other algorithms in terms of accuracy and efficiency. According to the simulations, results, analyses, discussions, and conclusions, it can be expressed that FLFBA have merits among the current optimization algorithms in the literature and worth applying to different problems as it is done in this thesis when we have applied the FLFBA to solve approximately an IVP, by selecting a specified example and after a comparison between the exact solutions, the algorithm outcomes, ABCA, BA, FPA and some numerical methods including Euler methods, Range-Kutta methods and predictor-corrector method results; FLFBA was found exponentially better by offering accurate solutions with smallest amount error.

4.7 Perspectives

FLFBA is an efficient optimization algorithm with a promising wide range of applications. It is important pointing out that the current results are mainly for the hybridization of the standard BA by using FLF and DE. It will be useful if further research can focus on the extension of the proposed methodology to optimize IVP by other variants of FLFBA. Ultimately, it can be expected that the proposed problem can be optimized by other MA as well. Based on the obtained results during this study, FLFBA having remarkable ability to solve a wide range of problems and highly nonlinear problems efficiently, it works well with complicated problems.

In future, there are deep studies on FLFBA that will give promising results such as the use of more diverse test function sets, more extensive comparison studies with wider range of existing algorithms; hence these comparisons will expose the qualities and limitations of all the algorithms. A further research on FLFBA that will improve the algorithm such as the parameter tuning, parameter control, speedup of coverage, using of more diverse parameters, more extensive comparison studies with more open sort of algorithms. . . , etc.

Furthermore, FLFBA should be tested in solving real problems with unknown search spaces to prove the effectiveness of these algorithms in practice. This researches make it possible to apply it to a wide range of problems in diverse applications and offer some advantages to FLFBA for some applications such as image compression, multi-objective optimization, and graph coloring. Furthermore, it is possible to extend the FLFBA to a discrete version so that it can solve combinatorial optimization problems as well as the proposition of binary versions of FLFBA which could be precious contributions. Despite the fact that FLFBA may be considered to optimize many engineering and industrial problems. All these extensions will be very useful. We hope that this thesis will inspire more active research in metaheuristics in the near future.

Bibliography

[1] Alcalá-Fdez, J., Sánchez, L., García, S., Del Jesus, M.J., Ventura, S., Garrell, J.M., Otero, J., Romero, C., Bacardit, J., Rivas, V.M., Fernández, J.C., Herrera, F., (2009). *KEEL: a software tool to assess evolutionary algorithms for data mining problems.* Soft Computing. 13-3(307–318), issn=1433-7479, doi="10.1007/s00500-008-0323-y.

[2] Amos, G. (2007). *Matlab, an introduction with application*s. John Willey and Sohn.

[3] Askarzadeh A., (2016). *A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm.* Computers and Structures. 169(Complete):1-12. doi:10.1016/j.compstruc. 2016.03.001.

[4] Arslanturk, C. (2009). *Correlation equations for optimum design of annular fins with temperature dependent thermal conductivity.* Heat Mass Transf. 45 (4) (2009) 519–525.

[5] Babaei, M. (2013). *A general approach to approximate solutions of nonlinear differential equations using particle swarm optimization*, Appl. Soft Comput. vol. 13, pp. 3354-3365.

[6] **Boudjemaa, R., Oliva, D., Ouaar, F., (2020). *Fractional Lévy Flight Bat Algorithm for Global Optimization.* International Journal of Bio-Inspired Computation. Inderscience publisher. Vol. 15, Issue. 2, pp. 100-112, DOI: 10.1504/IJBIC.2020.106441**.

[7] Boyce, W. E., DiPrima, R.C. (2001). *Elementary differential equation and boundary value problems.* John Wiley and Son.

[8] Bonabeau, E., Theraulaz, G. and Dorigo, M. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press. New York.

[9] Bhateja, A.K., Bhateja, A., Chaudhury, S., Saxena, P. (2015). *Cryptanalysis of vigenere cipher using cuckoo search*. Applied Soft Computing. 26:315-24. doi:https://doi.org/10.1016/j.asoc.2014.10.004.

[10] Blum, C., Roli, A. (2003). *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*. ACM Computing Surveys. 35 (3). 268–308.

[11] Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J., (2009). *A survey on metaheuristics for stochastic combinatorial optimization*. Natural Computing. 8 (2): 239–287. doi:10.1007/s11047-008-9098-4.

[12] Butcher, JC. (2008). *Numerical methods for ordinary differential equations*. John Wiley Sons.

[13] Cao, H., Kang, L., Chen Y. (2000) . *Evolutionary modeling of systems of ordinary differential equations with genetic programming*, Genet. Program. Evol. M., vol. 1, pp. 309-337.

[14] Cesari, L. (2012). *Optimization—theory and applications: problems with ordinary differential equations*. Springer Science Business Media, 17.

[15] Collatz, L. (1960). *Numerical treatment of differential equations*. Springer Verlag Berlin.

[16] Cos, S.B., kun, M.T. Atay., (2007). *Analysis of convective straight and radial fins with temperature-dependent thermal conductivity using variational iteration method with comparison with respect to finite element analysis*. Math. Probl. Eng. 2007. 1–15, ID 42072.

[17] Cos, S.B., kun, M.T. Atay., (2008). *Fin efficiency analysis of convective straight fins with temperature dependent thermal conductivity using variational iteration method*. Appl.Therm. Eng. 28 (2008) 2345–2352.

[18] Chiu, C.H., Chen, C.K. (2002). *Application of the decomposition method to thermal stresses in isotropic circular fins with temperature-dependent thermal conductivity.* Acta Mech. 157 (2002) 147–158.

[19] Couceiro, M., Sivasundaram, S. (2016). *Novel fractional order particle swarm optimization.* Applied Mathematical Computing. 283©:36-54. doi:10.1016/j.amc.2016.02.007.

[20] Chakri, A., Khelif, R., Benouaret, M., Yang, X.S. (2017). *New directional bat algorithm for continuous optimization problems.* Expert Systems with Applications. 69(Supplement C):159-75. doi:10.1016/j.eswa.2016.10.050.

[21] Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms.* PhD thesis. Politecnico di Milano. Italie.

[22] Derrac, J., Salvador, G., Molina, D., and Herrera, F. (2011). *A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms.* Swarm and Evolutionary Computation. 1-1(3 - 18).

[23] Domairry, G., Fazeli, M. (2009). *Homotopy analysis method to determine the fin efficiency of convective straight fins with temperature-dependent thermal conductivity.* Commun. Nonlinear Sci. 14 (2009) 489–499.

[24] Edelman, D.C. (2010). *Branding in the digital age: You're spending your money in the wrong places.* Harvard Business Review 2010;88(12):16-23.

[25] Fister, I., Fister, D., Yang, X.S. (2013). *A hybrid bat algorithm.* CoRR 2013. abs/1303.6310.

[26] Ganesan, T., Elamvazuthi, I., Shaari, K., Zilati, K., Vasant, P., (2013). *Swarm intelligence and gravitational search algorithm for multi-objective optimization of synthesis gas production.* Applied Energy. 103: 368–374. doi:10.1016/j.apenergy.2012.09.059.

[27] Glover, F., Kochenberger, G.A., (2003). *Handbook of metaheuristics*. 57. Springer. International Series in Operations Research & Management Science. ISBN 978-1-4020-7263-5.

[28] Glover, F., (1986). *Future paths for integer programming and links to artificial intelligence*. Computers and Operations Research. 13, 533–549 (1986).

[29] Goldberg D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st ed., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 1989. ISBN 0201157675.

[30] Guo, B-y., Yan, J-p. (2009). *Legendre–Gauss collocation method for initial value problems of second order ordinary differential equations*. Applied Numerical Mathematics, 59(6):1386-1408.

[31] Gilat, A. (2004). *Matlab: An Introduction with application*. John Wiley and Sons.

[32] Ghandomi, H. (2014). *Chaotic bat algorithm*. Computational Science 2014;5(2):224-32.

[33] Heidari, A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., (2019). *Harris hawks optimization: Algorithm and applications*. Future Generation Computer Systems. 97: 849–872. doi:10.1016/j.future.2019.02.028. ISSN 0167-739X.

[34] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press. ISBN 978-0-262-08213-6.

[35] Henrici, P. (1964). *Elements of Numerical Analysis*. Mc Graw-Hill. New York.

[36] Hirsch, M.W., Stephen, S., (1974). *Differential equations, dynamical systems, and linear algebra*. New York-London: Academic Press.

[37] Islam, T., Islam, M.E. and Ruhin, M.R. (2018). *An Analysis of Foraging and Echolocation Behavior of Swarm Intelligence Algorithms in Optimization: ACO, BCO and BA*. International Journal of Intelligence Science , 8, 1-27. https://doi.org/10.4236/ijis.2018.81001.

[38] Hull, T., Enright, W., Fellen, B., Sedgwick, A. (1972). *Comparing numerical methods for ordinary differential equations*. SIAM Journal on Numerical Analysis, 9(4): 603-637.

[39] Karaboga, D., (2010). *Artificial bee colony algorithm*. Scholarpedia. 5 (3): 6915. Bib code: 2010SchpJ.5.6915K. doi:10.4249/scholarpedia. 6915.

[40] Karaboga, D. (2005). *An Idea Based on Honey Bee Swarm for Numerical Optimization*. Technical Report-TR06. Erciyes University. Engineering Faculty. Computer Engineering Department.

[41] Kennedy, J., Eberhart, R. (1995). *Particle swarm optimization. In: Neural Networks*. Proceedings. IEEE International Conference on vol. 4. p. 1942-1948. doi:10.1109/ICNN.1995.488968.

[42] Kennedy, CA., Carpenter, MH. (2016). *Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations*. A Review.

[43] Khan, J., Zahoor, R., and Qureshi, I. (2009). *Swarm Intelligence for the Solution of Problems in Differential Equations*. Environmental and Computer Science, 2009. ICECS '09. Second International Conference on, Dec 2009, pp. 141–147.

[44] Kockler, N. (1994). *Numerical methods and scientific computing*. Clarendon Press. Oxford London.

[45] Kirkpatrick, S., Gelatt Jr.C.D. and Vecchi, M.P. (1983). *Optimization by Simulated Annealing*. Science. 220 (4598): 671–680. Bibcode:1983Sci.220.671K. CiteSeerX 10.1.1.123.7607. doi:10.1126/science.220.4598.671. PMID 17813860.

[46] Kirstukas, S.J., Bryden, K.M., and Ashlock, D.A. (2005). *A hybrid genetic programming approach for the analytical solution of differential equations*. International Journal of General Systems. Vol. 34, 2005, pp. 279–299.

[47] Kudryashov, N. A., Chmykhov, M. A. (2014). *Logistic function as solution of many nonlinear differential equations*. arXiv:1409.6896v1 [nlin.SI].

[48] Lambert, J.D., (1991). *Numerical method for ordinary systems of Initial value problems*. John Wiley and Sons. New York.

[49] Lagaris, I.E., Likas, A., and Fotiadis, D.I. (1998). *Artificial Neural Networks for Solving Ordinary and Partial Differential Equations*. IEEE Transactions on Neural Networks. Vol. 5, 1998, p. 987–1000.

[50] Lee, Z.Y., (2006), *Method of bilaterally bounded to solution Blasius equation using particle swarm optimization*, Appl. Math. Comput., vol. 179, pp. 779-786.

[51] Machado, J., (2002). *System modeling and control through fractional-order algorithms*. Nonlinear dynamics, chaos, control, and their applications to engineering sciences. 4:99-116.

[52] Machado T.J.A.T. (1997). *Analysis and design of fractional-order digital control systems*. Syst Anal Model Simul. 27(2-3):107-22.

[53] Mateescu G.D., (2005). *Optimization by using evolutionary algorithms with genetic acquisitions*. Romanian Journal of Economic Forecasting. No. 2/2005.

[54] Mantegna, R.N., (1994). *Fast, accurate algorithm for numerical simulation of lévy stable stochastic processes*. Phys Rev E. 49:4677-83. doi:10.1103/PhysRevE.49.4677.

[55] Meng, X., Gao, X., Liu, Y., Zhang, H., (2015). *A novel bat algorithm with habitat selection and doppler effect in echoes for optimization*. Expert Systems with Applications. 42(17-18):6350-64. doi:10.1016/j.eswa.2015.04.026.

[56] Mastorakis, E.N. (2006). *Unstable ordinary differential equations: solution via genetic algorithms and the method of Nelder-Mead*, In: Proc. of 6th WSEAS Int. Conf. on Systems Theory and Scientific Computation, 119, Elounda, Greece, pp. 1-6.

[57] Mirjalili, S., Lewis, A. (2016). *The whale optimization algorithm*. Adv Eng Softw 2016;95©:51-67. doi:10.1016/j.advengsoft.2016.01.008.

[58] MAZ Raja, J., Khan, I.Qureshi. (2011). *Swarm intelligence optimized neural networks for solving fractional differential equations*. International Journal of Innovative Computing, Information and Control, 7(2).

[59] Ogunrinde R. B., Fadugba S. E., Okunlola J. T. (2012). *On Some Numerical Methods for Solving Initial Value Problems in Ordinary Differential Equations*. IOSR Journal of Mathematics (IOSRJM). ISSN: 2278-5728 Volume 1. Issue 3. pp. 25-31.

[60] Oldham, K.B., Spanier, J. (1974). *The Fractional Calculus Theory and Applications of Differentiation and Integration to Arbitrary Order*. vol. 111 of Mathematics in Science and Engineering. Elsevier. 1974. doi:https: //doi.org/10.1016/S0076-5392(09)60219-8.

[61] **Ouaar, F.**, Khelil, N. (2019). *Optimization of Civil Engineering's Initial Value Problems by Particle Swarm Optimization Algorithm*. Int J S Res Civil Engg. January-February-2019; 3 (1): 01-07.

[62] **Ouaar, F.**, Khelil, N. (2018). *A Nature Inspired Algorithm based resolution of an Engineering's ODE*. Int J S Res Mech & Mtrls Engg. May-June-2018 2(2): 21-27.

[63] **Ouaar, F.**, Khelil, N. (2018). *Solving Initial Value Problems by Flower Pollination Algorithm*. American Journal of Electrical and Computer Engineering. Vol. 2, No. 2. pp. 31-36. doi: 10.11648/j.ajece.20180202.14.

[64] **Ouaar, F.**, Khelil, N. (2018). *Swarm Intelligence Algorithm for Solving Nonlinear Second Order Boundary Value Problems*. Proceedings of the 1st international conference on Artificial Intelligence and its Applications AIAP'2018 El Oued, Algeria, December 04–05, 2018, pp. 35-39.

[65] **Ouaar, F.**, Khelil, N. (2018). *On the application of the Bio-Inspired Algorithms in Optimization problems*. Proceedings of the 4^{th} Algerian Congress of Mathematicians (CMA-2018), Boumerdès, Algeria, May 12-13. pp. 203-205.

[66] Özkaya, E., Gauger, NR., Nemili, A. (2016). *Chaotic Behavior of Stiff ODEs and Their Derivatives: An Illustrative Example*. arXiv preprint arXiv:161003358.

[67] Polyanin, A.D., Zaitsev, V.F. (2003). *Handbook of exact solutions for ordinary differential equations (2nd ed)*. Boca Raton, Florida: Chapman & Hall/CRC. ISBN 1-58488-297-2.

[68] Qu, B.Y., Liang, J.J., Wang Z.Y., Chen, Q., and Suganthan, P.N. (2016). *Novel benchmark functions for continuous multimodal optimization with comparative results*. Swarm and Evolutionary Computation, vol 26. pp. 23 - 34.

[69] Rabei, E.M., Altarazi, I.M.A., Muslih S.I. and Baleanu, D. (2009). *Fractional wkb approximation*. Nonlinear Dynamics. 57(1):171-5. doi:10.1007/s11071-008-9430-7.

[70] Razmjooy, N., Ramezani, M., (2016). *Analytical solution for optimal control by the second kind Chebyshev polynomials expansion*. Iranian Journal of Science and Technology (Sciences).

[71] Rezaee, J.A. (2015). *Chaotic bat swarm optimization (cbso)*. Appl Soft Comput. 26©:523-30. doi:10.1016/j.asoc.2014.10.010.

[72] Seaton, T., Brown, G., and Miller, J.F. (2010). *Analytic Solutions to Differential Equations under Graph-Based Genetic Programming*. EuroGP LNCS, Vol. 6021, 2010, p. 232–243.

[73] Sadollah, A., Choi, Y.H.; Joong, H.K. (2015). *Metaheuristic optimization algorithms for approximate solutions to ordinary differential equations*. Conference paper in IEEE Congress on Evolutionary Computation (CEC 2015), At Sendai International Center, Sendai, Japan. DOI: 10.1109/CEC.2015.7256972

[74] Samuel, D.C. (1981). *Elementary numerical analysis (an algorithm approach)*. Third Edition. Mc Graw International Book Company.

[75] Salimi, H. (2014). *Stochastic fractal search*. Know-Based Syst 2015;75(C):1-18. doi:10.1016/j.knosys.2014.07.025.

[76] Sabatier, J., Agrawal, O., Machado, J., (2007). *Advances in Fractional Calculus: Theoretical Developments and Applications in Physics and Engineering*. Springer Netherlands. ISBN 9781402060410.

[77] Sörensen, Ke., Sevaux, M., Glover, F., (2017). *A History of Metaheuristics*. In Martí, Rafael. Panos, Pardalos; Resende, Mauricio (eds.). Handbook of Heuristics. Springer. ISBN 978-3-319-07123-7.

[78] Storn, R., Price, K., (1997). *Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces*. Global Optimization.11(4):341-59. doi:10.1023/A:1008202821328.

[79] Talbi, E-G., (2009). *Metaheuristics: from design to implementation*. Wiley. ISBN 978-0-470-27858-1.

[80] Tomoiagă, B., Chindriş, M., Sumper, A., Sudria-Andreu, A., Villafafila-Robles, R. (2013). *Pareto Optimal Reconfiguration of Power Distribution Systems Using a Genetic Algorithm Based on NSGA-II*. Energies. 2013. 6(3):1439–1455.

[81] Tsoularisa, A., Wallaceb,J. (2002). *Analysis of logistic growth models Author links open overlay panel*. Mathematical Bioscience Elsivier. Volume 179, Issue 1, Pages 21-55.

[82] Tsoulos, I.G., Lagaris, I.E. (2006). *Solving differential equations with genetic programming*. Genetic Programming and Evolvable Machines. Vol. 7, 2006, p. 33–54.

[83] Tsoulos, IG., Gavrilis, D., Glavas, E., (2009). *Solving differential equations with constructed neural networks*. Neurocomputing, 72(10):2385-2391.

[84] Viswanathan, G.M., Afanasyev, V., Buldyrev S.V., Murphy, E.J., Prince, P.A., Stanley, H.E. (1996). *Lévy flight search patterns of wandering albatrosses*. Nature. 381. doi:10.1038/381413a0.

[85] Wolpert, D.H., Macready W.G. (1997). *No free lunch theorems for optimization*. Trans Evol Comp 1997. 1(1):67-82. doi:10.1109/4235.585893.

[86] Xie, J., A Zhou, Y., A Chen, H. (2013). *A novel bat algorithm based on differential operator and lévy flight trajectory*. Computational Intelligence and Neuroscience. 2013. doi:R10.1155/2013/453812.

[87] Yang, X.S., (2011). *Metaheuristic optimization*, Scholarpedia, 6(8):11472 .

[88] Yang, X.S. (2010). *A New Metaheuristic Bat-Inspired Algorithm*. Berlin. Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-12538-6. p. 65-74. doi:10.1007/978-3-642-12538-6_6.

[89] Yang, X.S., Deb, S., (2009). *Cuckoo search via lévy flights.* In: 2009 World Congress on Nature Biologically Inspired Computing (NaBIC). p. 210-4. doi:10.1109/NABIC.2009.5393690.

[90] Yang, X.S. (2014). *Nature-Inspired Optimization Algorithms.* 1st ed. Amsterdam. The Netherlands, The Netherlands: Elsevier Science Publishers B.V. ISBN 0124167438. 9780124167438.

[91] Yilmaz, S., Ku cuksille, E.U., (2015). *A new modification approach on bat algorithm for solving optimization problems. Applied Soft Computing.* 28©:259-75. doi:10.1016/j.asoc.2014.11.029.

[92] Zarei, M.H., Davvari, Me., Kolahan, F., and Wong, KY. (2015). *Simultaneous Selection and Scheduling with Sequence-Dependent Set-up Times, Lateness Penalties, and Machine Availability Constraint: Heuristic Approaches.* International Journal of Industrial Engineering Computations, vol. 7 no. 1, pp. 147–160, 2016. DOI: 10.5267/j.ijiec.2015.7.001.

Appendix A

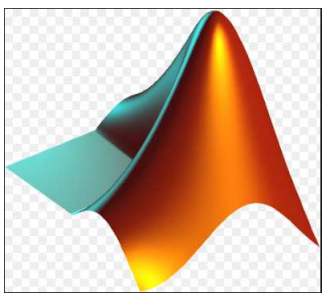
What is MATLAB?

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include [2, 31]:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, es-

Figure A.1: Matlab icon.



pecially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C or Fortran.

The Name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation. MATLAB has evolved over a period of years with input from many users.

In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis. MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-les) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

Appendix B

FLFBA tables

B.1 Benchmark functions

B.2 Multiple comparison tests tables

Table B.1: Benchmark functions

Category	**Name**	**Information gained**
Separable functions	Sphere function [F_1]	What is the optimal convergence rate of an algorithm?
	Ellipsoidal function [F_2]	Is symmetry (c.t. F_1) and separability (c.t. F_10) exploited
	Rastrigin function [F_3]	What is the effect of multi-modality?
	Büche-Rastrigin function [F_4]	What is the effects of asymmetry (c.t. F_3)?
	Linear slope [F_5]	Can the search go outside the initial convex hull of solutions into the domain boundary? Can the step size be increased accordingly?
Functions with low or moderate conditioning	Attractive Sector function [F_6]	What is the effect of highly asymmetric landscape (c.t. F_1)?
	Step Ellipsoidal function [F_7]	Does the search gets stuck on plateaus?
	Rosenbrock function, original [F_8]	Can the search follow a long path with D − 1 changes in the direction?
	Rosenbrock function rotated [F_9]	Can the search follow a long path with D − 1 changes in the direction without exploiting partial separability (c.t. F_8)?
Functions with high conditioning and unimodal	Ellipsoidal function [F_{10}]	What is the effect of rotation (c.t. F_2)?
	Discus function [F_{11}]	What is the effect of constraints (c.t. F_{10})?
	Bent cigar function [F_{12}]	Can the search continuously change its search direction?
	Sharp ridge function [F_{13}]	What is the effect of non-smoothness, non-differentiable ridge (c.t. F_{12})?
	Different powers function [F_{14}]	What is the effect of missing self-similarity?
Multimodal functions with adequate global structure	Rastrigin function [F_{15}]	What is the effect of non-separability for a highly multimodal function (c.t. F_3)?
	Weierstrass function [F_{16}]	Does ruggedness or a repetitive landscape deter the search behavior (c.t. F_{17})?
	Schaffers F7 function [F_{17}]	What is the effect of multimodality on a less regular function (c.t. F_{15})?
	Schaffers F7, moderately ill-conditioned [F_{18}]	What is the effect of ill-conditioning (c.t. F_{17})?
	Composite Griewank-Rosenbrock F8F2 [F_{19}]	What is the effect of high signal-to-noise ratio (c.t. F_7)?
Multimodal functions with weak global structure	Schwefel function [F_{20}]	What is the effect of a weak global structure (c.t. F_{17})?
	Gallagher's Gaussian 101-me peaks function [F_{21}]	Is the search effective without any global structure?
	Gallagher's Gaussian 21-hi peaks function [F_{22}]	What is the effect of higher condition (c.t. F_{21})?
	Katsuura function [F_{23}]	What is the effect of regular local structure on the global search?
	Lunacek bi-Rastrigin function [F_{24}]	Can the search behavior is local on the global scale but global on a local scale?

Table B.2: Average computational time of the different algorithms using 50 trials for the benchmarks function F_1 .. F_{24}.

Algorithm	$\mathbf{F_1}$	$\mathbf{F_2}$	$\mathbf{F_3}$	$\mathbf{F_4}$	$\mathbf{F_5}$	$\mathbf{F_6}$	$\mathbf{F_7}$	$\mathbf{F_8}$	$\mathbf{F_9}$	$\mathbf{F_{10}}$	$\mathbf{F_{11}}$	$\mathbf{F_{12}}$	$\mathbf{F_{13}}$	$\mathbf{F_{14}}$	$\mathbf{F_{15}}$	$\mathbf{F_{16}}$	$\mathbf{F_{17}}$	$\mathbf{F_{18}}$	$\mathbf{F_{19}}$	$\mathbf{F_{20}}$	$\mathbf{F_{21}}$	$\mathbf{F_{22}}$	$\mathbf{F_{23}}$	$\mathbf{F_{24}}$
										D – 10														
CSN	7.67	9.34	9.86	12.98	9.44	14.92	13.14	12.18	12.09	13.78	13.73	12.83	12.19	12.01	14.61	16.07	14.22	10.46	8.76	9.93	11.86	11.50	9.36	8.53
FDPSO	12.13	27.58	19.22	19.79	11.62	19.64	8.41	16.71	18.14	21.57	8.69	12.86	13.15	11.48	20.41	10.24	14.80	11.62	5.90	14.42	24.97	25.07	10.70	8.16
NBA	2.11	2.75	3.18	3.27	1.72	2.91	2.40	2.27	2.44	2.90	2.79	2.52	2.33	2.37	3.21	3.58	3.17	3.03	2.26	2.61	4.01	3.51	3.87	2.40
ACO	3.58	4.38	4.61	5.04	2.50	4.48	3.79	3.48	3.57	4.25	4.23	3.88	3.46	3.57	4.96	5.20	4.56	4.41	3.54	3.99	5.27	4.87	4.43	3.67
MFO	4.28	4.18	3.99	4.15	2.59	4.27	4.32	3.40	1.74	2.18	2.27	1.94	1.69	1.81	2.53	2.78	2.34	2.35	1.73	1.98	3.06	2.69	2.22	1.77
SFLA	14.86	14.85	14.75	15.22	12.41	15.34	15.70	12.60	12.40	15.74	16.57	12.78	11.25	13.22	14.82	16.24	14.01	14.18	15.80	13.11	17.16	16.03	16.93	13.48
FLFBA	4.11	6.77	7.05	6.90	5.85	8.58	7.37	6.78	7.45	8.46	5.56	4.20	3.87	5.71	5.39	5.90	5.16	5.00	4.05	4.53	6.44	5.84	11.19	6.93
										D – 20														
CSN	15.41	19.30	21.06	21.89	15.15	19.47	18.12	16.62	17.00	20.46	20.16	18.06	17.88	16.89	21.05	22.97	20.79	21.00	16.85	19.62	25.57	23.88	19.80	18.22
FDPSO	18.85	48.92	32.50	33.39	16.15	33.77	14.10	22.58	20.10	30.80	17.76	22.04	20.79	20.64	31.78	19.59	25.38	21.50	12.59	21.66	38.35	34.70	13.55	14.92
NBA	4.11	5.27	5.81	5.92	3.13	5.33	4.30	4.08	4.07	5.37	5.27	4.48	3.94	4.21	6.06	7.47	6.08	5.96	4.33	4.81	7.25	6.38	8.57	4.27
ACO	12.04	11.83	11.24	11.35	5.87	10.68	9.57	8.82	9	10.68	10.49	9.63	8.72	8.94	11.44	12.71	10.97	10.85	9.27	9.61	12.69	11.78	11.08	8.92
MFO	2.92	4.21	4.67	4.83	2.22	4.48	3.66	3.13	3.26	4.33	4.21	3.62	3.07	3.22	4.83	5.72	4.60	4.52	3.28	3.66	5.84	5.14	4.43	3.30
SFLA	27.64	32.95	34.59	34.50	24.55	32.14	32.14	25.11	24.49	32.25	34.61	25.51	24.14	27.79	33.81	35.06	29.19	29.19	31.82	30.60	42.00	39.10	32.58	27.17
FLFBA	11.40	19.17	14.79	16.7	8.76	16.82	15.33	15.00	18.62	18.43	18.87	15.92	20.52	11.88	16.25	20.32	24.63	17.32	9.09	9.51	14.04	14.83	18.88	10.57
										D – 40														
CSN	32.71	42.03	45.92	46.82	35.60	40.45	38.40	35.20	36.03	43.32	42.77	38.01	34.87	35.19	46.59	52.62	46.1	45.76	35.94	41.73	58.64	49.63	42.26	37.01
FDPSO	28.35	70.98	48.53	48.21	21.30	52.25	22.67	32.37	30.68	48.40	34.88	33.61	33.08	32.03	50.65	46.96	47.38	41.65	23.79	34.15	67.42	54.93	28.65	27.83
NBA	7.01	10.75	12.58	12.64	7.28	10.92	9.00	7.69	8.08	11.30	11.29	9.15	7.92	8.56	13.20	20.50	11.89	11.83	8.04	9.32	15.73	12.78	17.46	8.59
ACO	23.68	28.73	30.42	30.62	16.74	28.70	26.52	24.63	25.25	29.34	29.19	26.57	24.73	25.46	31.60	37.38	30.25	30.15	25.35	26.50	34.47	31.42	30.70	25.31
MFO	5.63	8.77	9.88	10.10	4.35	8.77	7.43	6.28	6.59	9.23	9.04	7.38	6.24	6.67	10.54	14.10	9.87	9.85	6.67	7.38	13.21	10.72	9.87	6.83
SFLA	59.91	71.40	76.01	75.77	53.66	72.66	69.47	54.53	56.20	69.08	70.77	57.26	52.53	59.69	70.98	78.38	66.55	65.54	66.85	64.00	88.60	79.58	65.68	59.27
FLFBA	15.32	19.00	21.13	21.18	10.13	18.91	16.97	20.28	26.82	22.85	29.75	26.60	23.02	25.27	34.83	59.88	30.17	31.09	23.40	25.42	39.03	32.27	59.71	24.22

Table B.3: Average Rankings of the algorithms

Algorithm	Friedman	Aligned Friedman	Quade
D – 10 CASES			
CS	4.9792	109.4375	5.02167
FDPSO	4.5625	102.9792	4.5716
NBA	2.2291	51.1041	2.1183
ACO	3.8125	80.8125	3.8066
MFO	3.1875	63.0625	2.9116
SFLA	5.666	112.3334	5.9183
FLFBA	3.5624	71.7708	3.6516
D – 20 CASES			
CS	5.0416	107.8333	5.0633
FDPSO	4.6667	95.8333	4.6733
NBA	2.4583	62.9166	2.380
ACO	3.75	81.9583	3.7066
MFO	2.5000	46.3333	2.0466
SFLA	5.9583	110.9166	6.4366
FLFBA	3.625	85.7083	3.6933
D – 40 CASES			
CS	5.29166	107.9166	5.2866
FDPSO	5.2083	98.6249	5.1233
NBA	2.9166	74.3333	2.71
ACO	2.0	57.6666	2.1433
MFO	2.3333	47.5416	1.9100
SFLA	6.125	108.875	6.4866
FLFBA	4.1245	96.5416	4.339

B.3 Post-hoc procedures tables

Table B.4: Contrast Estimation

D – 10 CASES

	CS	FDPSO	NBA	ACO	MFO	SFLA	FLFBA
CS	0	-0.9582	-7.124	-2.193	-4.534	1.631	-2.774
FDPSO	0.9582	0	-6.165	-1.235	-3.576	2.589	-1.815
NBA	7.124	6.165	0	4.931	2.589	8.754	4.350
ACO	2.193	1.235	-4.931	0	-2.341	3.823	-0.5806
MFO	4.534	3.576	-2.589	2.341	0	6.165	1.761
SFLA	-1.631	-2.589	-8.754	-3.823	-6.165	0	-4.404
FLFBA	2.774	1.815	-4.350	0.5806	-1.761	4.404	0

D – 20 CASES

	CS	FDPSO	NBA	ACO	MFO	SFLA	FLFBA
CS	0.000	-5.059	-19.68	-7.410	-28.00	9.945	-9.221
FDPSO	5.059	0.000	-14.62	-2.350	-22.94	15.00	-4.161
NBA	19.68	14.62	0.000	12.27	-8.321	29.62	10.46
ACO	7.410	2.350	-12.27	0.000	-20.59	17.35	-1.811
MFO	28.00	22.94	8.321	20.59	0.000	37.95	18.78
SFLA	-9.945	-15.00	-29.62	-17.35	-37.95	0.000	-19.17
FLFBA	9.221	4.161	-10.46	1.811	-18.78	19.17	0.000

D – 40 CASES

	CS	FDPSO	NBA	ACO	MFO	SFLA	FLFBA
CS	0.000	-34.32	-86.68	-177.6	-157.2	42.20	-32.72
FDPSO	34.32	0.000	-52.36	-143.3	-122.9	76.52	1.598
NBA	86.68	52.36	0.000	-90.97	-70.56	128.9	53.96
ACO	177.6	143.3	90.97	0.000	20.41	219.8	144.9
MFO	157.2	122.9	70.56	-20.41	0.000	199.4	124.5
SFLA	-42.20	-76.52	-128.9	-219.8	-199.4	0.000	-74.92
FLFBA	32.72	-1.598	-53.96	-144.9	-124.5	74.92	0.000

Table B.5: Holm -Hochberg - Hommel (H-H-H)/ Holland / Rom / Finner / Li Table for $\alpha = 0.05$ in $D - 10$

i	**algorithm**	**z** $= \frac{\mathbf{R_0 - R_i}}{\mathbf{SE}}$	**p**	**H-H-H**	**Holland**	**Rom**	**Finner**	**Li**
			FRIEDMAN					
6	SFLA	5.5122	3.5425E-8	0.0083	0.00851	0.0087	0.0085	0.0461
5	CS	4.4098	1.0346E-5	0.01	0.0102	0.0105	0.0169	0.0461
4	FDPSO	3.7416	1.8281E-4	0.0125	0.0127	0.0131	0.0253	0.0461
3	ACO	2.5389	0.0111	0.01666	0.0169	0.0166	0.0336	0.0461
2	FLFBA	2.1380	0.0325	0.025	0.0253	0.025	0.0418	0.0461
1	MFO	1.5367	0.1243	0.05	0.05	0.05	0.05	0.05
			ALIGNED FRIEDMAN					
6	SFLA	4.360	1.2973E-5	0.0083	0.0085	0.0087	0.0085	0.0318
5	CS	4.1543	3.2625E-5	0.01	0.0102	0.0105	0.0169	0.0318
4	FDPSO	3.6943	2.2042E-4	0.0125	0.0127	0.0131	0.0253	0.0318
3	ACO	2.1157	0.0343	0.0166	0.0169	0.0166	0.0336	0.0318
2	FLFBA	1.4718	0.1410	0.025	0.0253	0.025	0.04184	0.0318
1	MFO	0.8516	0.3944	0.05	0.05	0.05	0.05	0.05
			QUADE					
6	SFLA	3.0777	0.002	0.0083	0.0085	0.0087	0.0085	0.0252
5	CS	2.3514	0.0186	0.01	0.0102	0.0105	0.0169	0.0252
4	FDPSO	1.9870	0.0469	0.0125	0.0127	0.0131	0.0253	0.0252
3	ACO	1.3674	0.1714	0.0166	0.0169	0.0166	0.0336	0.0252
2	FLFBA	1.2418	0.2142	0.025	0.0253	0.025	0.0419	0.0252
1	MFO	0.6425	0.5205	0.05	0.05	0.05	0.05	0.05

Table B.6: Holm -Hochberg - Hommel (H-H-H)/ Holland/ Rom / Finner / Li Table for $\alpha = 0.05$ in $D - 20$

i	**algorithm**	**z** $= \frac{\mathbf{R_0 - R_i}}{\mathbf{SE}}$	**p**	**H-H-H**	**Holland**	**Rom**	**Finner**	**Li**
			FRIEDMAN					
6	SFLA	5.6124	1.9944E-8	0.0083	0.0085	0.0087	0.0085	0.0028
5	CS	4.1425	3.4346E-5	0.01	0.0102	0.0105	0.0169	0.0028
4	FDPSO	3.5412	3.9829E-4	0.0125	0.0127	0.0131	0.0253	0.0028
3	ACO	2.07127	0.0383	0.0166	0.0169	0.0166	0.0336	0.0028
2	FLFBA	1.8708	0.0613	0.025	0.0253	0.025	0.0418	0.0028
1	MFO	0.0668	0.9467	0.05	0.05	0.05	0.05	0.05
			ALIGNED FRIEDMAN					
6	SFLA	4.5994	4.2365E-6	0.0083	0.0085	0.0087	0.0085	0.0401
5	CS	4.3798	1.1876E-5	0.01	0.0102	0.0105	0.0169	0.0401
4	FDPSO	3.5252	4.2310E-4	0.0125	0.0127	0.0131	0.0253	0.0401
3	FLFBA	2.8041	0.0050	0.0166	0.0169	0.0166	0.0336	0.0401
2	ACO	2.5371	0.0111	0.025	0.0253	0.025	0.0418	0.0401
1	NBA	1.1810	0.2375	0.05	0.05	0.05	0.05	0.05
			QUADE					
6	SFLA	3.5555	3.7716E-4	0.0083	0.0085	0.0087	0.0085	0.0112
5	CS	2.4432	0.0145	0.01	0.0102	0.0101	0.0169	0.0112
4	FDPSO	2.1274	0.0333	0.0125	0.0127	0.0131	0.0253	0.0112
3	ACO	1.3444	0.1787	0.0166	0.0169	0.0166	0.0336	0.0112
2	FLFBA	1.3336	0.1823	0.025	0.0253	0.025	0.0418	0.0112
1	NBA	0.2699	0.7871	0.05	0.05	0.05	0.05	0.05

Table B.7: Holm -Hochberg - Hommel (H-H-H)/ Holland/ Rom / Finner / Li Table for $\alpha = 0.05$ in $D - 40$

i	**algorithm**	**z** $= \frac{\mathbf{R_0 - R_i}}{\mathbf{SE}}$	**p**	**H-H-H**	**Holland**	**Rom**	**Finner**	**Li**
			FRIEDMAN					
6	SFLA	6.6147	3.7226E-11	0.0083	0.0085	0.0087	0.0085	0.0214
5	CS	5.2784	1.3030E-7	0.01	0.0102	0.0105	0.0169	0.0214
4	FDPSO	5.1447	2.6783E-7	0.0125	0.0127	0.0131	0.0253	0.0214
3	FLFBA	3.4075	6.5541E-4	0.0166	0.0169	0.0166	0.0336	0.0214
2	NBA	1.4699	0.1415	0.025	0.0253	0.025	0.0418	0.0214
1	MFO	0.5345	0.5929	0.05	0.05	0.05	0.05	0.05
			ALIGNED FRIEDMAN					
6	SFLA	4.3679	1.2540E-5	0.0083	0.0085	0.0087	0.0085	0.0278
5	CS	4.2997	1.7101E-5	0.01	0.0102	0.0105	0.0169	0.0278
4	FDPSO	3.6379	2.7476E-4	0.0125	0.0127	0.0131	0.0253	0.0278
3	FLFBA	3.4896	4.8369E-4	0.01666	0.0169	0.0166	0.0336	0.0278
2	NBA	1.9080	0.05638	0.025	0.0253	0.025	0.0418	0.0278
1	ACO	0.7210	0.4708	0.05	0.05	0.05	0.05	0.05
			QUADE					
6	SFLA	3.7067	2.0993E-4	0.0083	0.0085	0.0087	0.0085	0.0078
5	CS	2.7348	0.0062	0.01	0.0102	0.0105	0.0169	0.0078
4	FDPSO	2.6025	0.0092	0.0125	0.0127	0.0131	0.0253	0.0078
3	FLFBA	1.9681	0.04905	0.0166	0.0169	0.0166	0.0336	0.0078
2	NBA	0.6479	0.5170	0.025	0.0253	0.025	0.0418	0.0078
1	ACO	0.1889	0.8501	0.05	0.05	0.05	0.05	0.05

Table B.8: Adjusted p-values (FRIEDMAN / ALIGNED FRIEDMAN / QUADE) in $D - 10$

FRIEDMAN

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	3.5424E-8	2.1254E-7	2.1254E-7	2.1254E-7	2.1254E-7
2	CS	1.0346E-5	6.2076E-5	5.1730E-5	5.1730E-5	5.1730E-5
3	FDPSO	1.8281E-4	0.0010	7.3124E-4	7.3124E-4	7.3124E-4
4	ACO	0.0111	0.0667	0.03335	0.03335	0.03335
5	FLFBA	0.0325	0.1950	0.0650	0.0650	0.0650
6	MFO	0.1243	0.7461	0.1243	0.1243	0.1243

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	3.5424E-8	2.1254E-7	2.0210E-7	2.1254E-7	4.0455E-8
2	CS	1.0346E-5	5.1729E-5	4.9195E-5	3.1038E-5	1.1815E-5
3	FDPSO	1.8281E-4	7.3104E-4	6.9725E-4	3.6558E-4	2.0872E-4
4	ACO	0.0111	0.0329	0.0333	0.0166	0.0125
5	FLFBA	0.0325	0.0639	0.0650	0.0388	0.0357
6	MFO	0.1243	0.1243	0.1243	0.1243	0.1243

ALIGNED FRIEDMAN

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	1.2973E-5	7.7840E-5	7.7840E-5	7.7840E-5	7.7840E-5
2	CS	3.2625E-5	1.9575E-4	1.6312E-4	1.6312E-4	1.63125E-4
3	FDPSO	2.2042E-4	0.0013	8.8170E-4	8.8170E-4	8.8170E-4
4	ACO	0.0343	0.2062	0.1031	0.1031	0.1031
5	FLFBA	0.1410	0.8464	0.2821	0.2821	0.2821
6	MFO	0.3944	2.3664	0.3944	0.3944	0.3944

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	1.2973E-5	7.7837E-5	7.4013E-5	7.7837E-5	2.1422E-5
2	CS	3.2625E-5	1.6311E-4	1.5513E-4	9.7873E-5	5.3871E-5
3	FDPSO	2.2042E-4	8.8141E-4	8.4071E-4	4.4080E-4	3.6385E-4
4	ACO	0.0343	0.0995	0.1031	0.0511	0.0537
5	FLFBA	0.1410	0.2622	0.2821	0.1668	0.1889
6	MFO	0.3944	0.3944	0.3944	0.3944	0.3944

QUADE

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	0.0020	0.0125	0.0125	0.0125	0.0125
2	CS	0.0186	0.1121	0.0934	0.0934	0.0934
3	FDPSO	0.0469	0.2815	0.1876	0.1876	0.1876
4	ACO	0.1714	1.0289	0.5144	0.4285	0.3429
5	FLFBA	0.2142	1.2856	0.5144	0.4285	0.4285
6	MFO	0.5205	3.1231	0.5205	0.5205	0.5205

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	0.0020	0.0124	0.0119	0.0124	0.0043
2	CS	0.0186	0.0900	0.0889	0.0550	0.0375
3	FDPSO	0.0469	0.1748	0.1789	0.0916	0.0891
4	ACO	0.1714	0.4312	0.4285	0.2458	0.2634
5	FLFBA	0.2142	0.4312	0.4285	0.2512	0.3088
6	MFO	0.5205	0.5205	0.5205	0.5205	0.5205

Table B.9: Adjusted p-values (FRIEDMAN / ALIGNED FRIEDMAN / QUADE) in $D - 20$

FRIEDMAN

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	1.9944E-8	1.1966E-7	1.1966E-7	1.1966E-7	1.1966E-7
2	CS	3.4346E-5	2.0607E-4	1.7173E-4	1.7173E-4	1.7173E-4
3	FDPSO	3.9829E-4	0.0023	0.0015	0.0015	0.0015
4	ACO	0.0383	0.2299	0.1149	0.1149	0.0920
5	FLFBA	0.0613	0.3682	0.1227	0.1227	0.1227
6	MFO	0.9467	5.6803	0.9467	0.9467	0.9467

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	1.9944E-8	1.1966E-7	1.1378E-7	1.1966E-7	3.7438E-7
2	CS	3.4346E-5	1.7172E-4	1.6331E-4	1.0303E-4	6.4433E-4
3	FDPSO	3.9829E-4	0.0015	0.0015	7.9642E-4	0.0074
4	ACO	0.03833	0.1106	0.1149	0.0569	0.4184
5	FLFBA	0.0613	0.1189	0.1227	0.0731	0.5353
6	MFO	0.9467	0.9467	0.9467	0.9467	0.9467

ALIGNED FRIEDMAN

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	4.2365E-6	2.5419E-5	2.5419E-5	2.5419E-5	2.5419E-5
2	CS	1.1876E-5	7.1259E-5	5.9382E-5	5.9382E-5	5.9382E-5
3	FDPSO	4.2310E-4	0.0025	0.0016	0.0016	0.0016
4	FLFBA	0.0050	0.0302	0.0151	0.0151	0.0151
5	ACO	0.0111	0.0670	0.0223	0.0223	0.0223
6	NBA	0.2375	1.4255	0.2375	0.2375	0.2375

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	4.2365E-6	2.5419E-5	2.4169E-5	2.5419E-5	5.5568E-6
2	CS	1.1876E-5	5.9381E-5	5.6472E-5	3.56295E-5	1.5577E-5
3	FDPSO	4.2310E-4	0.0016	0.0016	8.4602E-4	5.5465E-4
4	FLFBA	0.0050	0.0150	0.0151	0.0075	0.0065
5	ACO	0.0111	0.0222	0.0223	0.0133	0.0144
6	NBA	0.2375	0.2375	0.2375	0.2375	0.2375

QUADE

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	3.7716E-4	0.0022	0.0022	0.0022	0.0022
2	CS	0.0145	0.0873	0.0727	0.0727	0.0727
3	FDPSO	0.0333	0.2003	0.1335	0.1335	0.1335
4	ACO	0.1787	1.0727	0.5363	0.3646	0.3575
5	FLFBA	0.1823	1.0938	0.5363	0.3646	0.3646
6	NBA	0.7871	4.7230	0.7871	0.7871	0.7871

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	3.7716E-4	0.0022	0.0021	0.0022	0.0017
2	CS	0.0145	0.0706	0.0692	0.0430	0.0640
3	FDPSO	0.0333	0.1270	0.1273	0.0656	0.1356
4	ACO	0.1787	0.4461	0.3646	0.2558	0.4565
5	FLFBA	0.1823	0.4461	0.3646	0.2558	0.4613
6	NBA	0.7871	0.7871	0.7871	0.7871	0.7871

Table B.10: Adjusted p-values (FRIEDMAN / ALIGNED FRIEDMAN / QUADE) in $D - 40$

FRIEDMAN

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	3.7226E-11	2.2335E-10	2.2335E-10	2.2335E-10	2.2335E-10
2	CS	1.3030E-7	7.8185E-7	6.5154E-7	6.5154E-7	6.5154E-7
3	FDPSO	2.6783E-7	1.6070E-6	1.0713E-6	1.0713E-6	1.0713E-6
4	FLFBA	6.5541E-4	0.0039	0.0019	0.0019	0.0019
5	NBA	0.1415	0.8494	0.2831	0.2831	0.2831
6	MFO	0.5929	3.5578	0.5929	0.5929	0.5929

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	3.7226E-11	2.2335E-10	2.1237E-10	2.2335E-10	9.1461E-11
2	CS	1.3030E-7	6.5154E-7	6.1961E-7	3.9092E-7	3.2015E-7
3	FDPSO	2.6783E-7	1.0713E-6	1.0215E-6	5.3567E-7	6.5804E-7
4	FLFBA	6.5541E-4	0.0019	0.0019	9.8296E-4	0.0016
5	NBA	0.1415	0.2631	0.2831	0.1673	0.2580
6	MFO	0.5929	0.5929	0.5929	0.5929	0.5929

ALIGNED FRIEDMAN

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	1.2540E-5	7.5243E-5	7.5243E-5	7.5243E-5	6.2702E-5
2	CS	1.7101E-5	1.0260E-4	8.5506E-5	8.5506E-5	8.5506E-5
3	FDPSO	2.7476E-4	0.0016	0.0010	0.0010	9.6738E-4
4	FLFBA	4.8369E-4	0.0029	0.0014	0.0014	0.0014
5	NBA	0.0563	0.3383	0.1127	0.1127	0.1127
6	ACO	0.4708	2.8251	0.4708	0.4708	0.4708

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	1.2540E-5	7.5241E-5	7.1544E-5	7.5241E-5	2.3699E-5
2	CS	1.7101E-5	8.5503E-5	8.1315E-5	7.5241E-5	3.2318E-5
3	FDPSO	2.7476E-4	0.0010	0.0010	5.4945E-4	5.1900E-4
4	FLFBA	4.8369E-4	0.0014	0.0014	7.2544E-4	9.1328E-4
5	NBA	0.0563	0.1095	0.1127	0.0672	0.0963
6	ACO	0.4708	0.4708	0.4708	0.4708	0.4708

QUADE

i	algorithm	unadjusted p	p_{Bonf}	p_{Holm}	p_{Hoch}	p_{Homm}
1	SFLA	2.0993E-4	0.0012	0.0012	0.0012	0.0012
2	CS	0.0062	0.0374	0.0312	0.0312	0.0249
3	FDPSO	0.0092	0.0555	0.0370	0.0370	0.0370
4	FLFBA	0.0490	0.2943	0.1471	0.1471	0.1471
5	NBA	0.5170	3.1021	1.0340	0.8501	0.8501
6	ACO	0.8501	5.1006	1.0340	0.8501	0.8501

i	algorithm	unadjusted p	p_{Holl}	p_{Rom}	p_{Finn}	p_{Li}
1	SFLA	2.0993E-4	0.0012	0.0011	0.0012	0.0013
2	CS	0.0062	0.0308	0.0296	0.0186	0.0399
3	FDPSO	0.0092	0.0365	0.0352	0.0186	0.0581
4	FLFBA	0.0490	0.1400	0.1471	0.0726	0.2465
5	NBA	0.5170	0.7667	0.8501	0.5824	0.7752
6	ACO	0.8501	0.8501	0.8501	0.8501	0.8501

Appendix C

Statistical tests

C.1 Multiple comparison tests

Generally pairwise comparisons, which are statistical method requiring only that the observations in a pair is ordered, is not influenced by any external factor, whereas in a multiple comparison, the set of the chosen algorithms can determine the results of the analysis [22]. Therefore, a pairwise comparison test, such as Wilcoxon's rank sum test, should not be used to conduct various comparisons involving a set of algorithms, because the Family-Wise Error Rate (FWER), is not controlled [1].

Multiple comparison procedures are designed to allow fixing the FWER before performing the analysis and take into account all the influences that can exist within the set of results for each algorithm. In this part three well- known multiple comparison tests are presented which are: The Friedman test, the Friedman Aligned Ranks test and the Quade test that should be employed in order to distinguish whether significant differences happen between inspected algorithms. Additionally, these tests rank the algorithms from the best performing to the least fortunate one.

The Friedman test is a nonparametric complement of the parametric two-way analysis of variance. The objective of this test is to verify whether there are noteworthy contrasts among the selected algorithms. This test decides the ranks of the algorithms for every individual information and balances the average rank of algorithms. The null hypothesis expresses that all the algorithms execute unvaryingly so their ranks should be the same. By means of this hypothesis the Friedman statistic is distributed via a chi-square distri-

bution with $(k-1)$ degrees of freedom, when n and k are sufficiently enormous ($n > 10$ and $k > 5$).

Since the Friedman test is just allowed for intra-set examinations which may be considered as a drawback especially if there is a little number of compared algorithms because in this case the inter-set comparisons cannot be significant so **the Friedman Aligned Ranks test** is used. In this technique the average execution given by all algorithms over individual data sets is equal to the location value. At that point, the distinction between the performance accomplished by an algorithm and the location value is registered. This progression is rehashed for every combination of algorithms and data sets. So acquired contrasts are considered as aligned observations and carry on their characteristics regarding to the data set and the combination of algorithms to which they belong. Then they are positioned from

1 to kn with respect to one another and the ranks assigned to the aligned observations. The test statistic is evaluated through a chi-square distribution with $(k-1)$ degrees of freedom [22].

The Iman and Davenports test are metric derived from the Friedmans statistic given that this last metric produces a conservative undesirable effect. Iman and Davenport proved that Friedmans chi-square distribution is conventional and developed an improved statistic which is distributed via the F-distribution with $(k-1)$ and $(k-1)(n-1)$ degrees of freedom [1].

The third test for multivariate comparisons is **the Quade test**, it's considered as an elective trial of Friedman by means of the difficulty contemplations. Toward the starting the Quade test has an analogous process to Friedman tests, establishes the ranks of the algorithms for every individual data set. After that, utilizing the original standards of algorithms performance, ranks are allowed to the data sets depending on the size of the sample range for each data set (the sample range is the difference between the biggest and the littlest perceptions for a given data set). Therefore we get n sample ranges, one for every datum set. The data set with the littlest range acquires the rank of 1, the following gets rank 2, and so on. If there should be an occurrence of ties average ranks are allocated. Lastly, the data set rank is increased by the distinction among the rank for

this data set and the average rank over all data sets. The final formula for Quade test statistics is distributed according to the F-distribution with $(k-1)$ and $(k-1)(n-1)$ degrees of freedom where k is the number of the tested algorithms and n is the number of the considered problems. Both, the Frideman Aligned Rank test and the Quade test, can be used under the same circumstances as the Friedman test. The differences in power between them are unknown, but users are encouraged to use these tests when the number of algorithms to be compared is low [1].

The Contrast Estimation of medians is a strategy to appraise the contrasts between numerous algorithms. This technique is entirely suggested able in the event that we expect that the global performance is reflected by the extents of the distinctions between the performances of the algorithms. It can be utilized to gage the contrast involving the performance of two algorithms. It supposes that the expected contrasts linking algorithms performances are the equivalent crosswise problems. Consequently, the execution of algorithms is reflected by the magnitudes of the differences between them in each domain. The enthusiasm of this test lies in evaluating the contrast between medians of samples of results taking into account all pairwise comparisons. The test acquires a quantitative contrast processed through medians between two algorithms over several problems. These estimators can be comprehended as a progressed performance measure. Despite the fact that this test can't furnish a probability of error related with the dismissal of the null hypothesis of equality, it is particularly helpful to assess by how far an algorithm outperforms another one.

C.2 Post-hoc procedures

If the statistical significance is rejected; at that point the scientist may continue to achieve Post-hoc procedures to point out which pair of algorithms varies fundamentally. A Post-hoc procedures in multiple comparison tests are focused on the comparison between a control method, which is usually the proposed method, and a set of algorithms used in the empirical study. Particularly when consider that Friedman, ImanDavenport, Friedman Aligned, and Quade tests can simply significant differences in excess of the complete multiple comparison. This negative aspect makes it inadequate to make precise com-

parisons between the selected algorithms. Then, the utilization of a Post-hoc test can prompt acquiring a p-value decides the level of dismissal of every speculation that grant to set up which algorithms are significantly better or worse. This type of comparison involves a control method, defined as the most interesting algorithm for the researcher of the experimental study, which its performance will be contrasted in a $1 \times n$ comparison against other algorithms selected in the study. Three classical Post-hoc procedures have been used in multiple comparisons tests and also valid in $n \times n$ comparisons that are Bonferroni-Dunn, Holm and Hochberg also Hommel, Rom, Finner tests can be suitable [22].

Bonferroni-Dunn test: It's a one-step procedure using a single step in the adjustment of the value, it controls the FWER by dividing the value of by the number of comparisons performed $(k-1)$ where k is the number of treated algorithms. The test is the simplest procedure in $1 \times n$ but has little efficiency when it comes to results. For this reason other procedures such as Holms or Hochbergs are preferred.

Holms method: It's a step-down procedure, adjusting the value of a step down manner, it's considered as a best-performing test and strongly recommended in rigorous comparison. Holms procedure can always be considered better and more powerful than Bonferroni Dunns one, because it appropriately controls the FWER.

Holland: It's a step down procedures also adjusts the value of a step-down manner, as Holms method does.

Hochbergs method: It's a step up procedures by adjusting the value of in a step up way. It works by comparing the largest p-value with α, the next largest with $\alpha/2$, the next with $\alpha/3$, and so forth until it finds a hypothesis it can reject. All hypotheses with smaller p-values are then rejected as well. The Hochbergs method is more powerful than Holms although it may under some circumstances exceed the FWER and can reject more hypothesis than the Holms method. The differences reported between the two procedures in practice are rather small. It is recommended to use this test together with the Holms method.

Hommel: Its a step up procedures, more complicated to compute and understand.

Rom: It's a step up procedures considered as an improvement of the Hochbergs proce-

dure.

Hochberg, Finner and Li are more recommended Post-hoc test to be used due to their trade-o between simplicity and power. The power of the Li test which is a two-step rejection procedure is highly influenced by the first p-value of the family. When the p-value is lower than 0.5, the test will perform very well [22] [1].

C.3 Unadjusted p-values

To demonstrate the contrasts between the three tests and their individual approximations for getting the p-value (additionally named unadjusted p-values), of each speculation, we will compute the unadjusted p-values for the considered algorithms as it provides information about whether a statistical hypothesis test is significant or not, and it also indicates something about how significant the difference is. The smaller the p-value means the stronger the evidence against the null hypothesis.

Many distinctions can be elucidated; Friedman test demonstrates a lower control than the Friedman Aligned test (the unadjusted p-values are extensively lower). In a multiple comparison tests the p-values are not suitable because it does not reflect on the rest of comparisons going to the family, it simply symbolizes the probability error of a certain comparison. One way to solve this problem is to report Adjusted P-Values (APVs) which takes into account that multiple tests are conducted. APVs care for this issue. They are appropriate to be utilized as they present more information in a statistical analysis. They assume the total family error, also they can be compared directly with any chosen significance level. The use of APVs is very counsel due to the fact that they provide more information in a statistical analysis [1].

Appendix D

FLFBA matlab code

```
% Main programs
function [bestX,FunMin,time] = FLFBA_OPT(FUN, DIM, ftarget, maxfunevals)
% Display help
%help NBA.m
% set the default parameters
tic;
M = maxfunevals;
pop = 30;
dim = DIM;
gama = 0.9; %0.9
alpha = 0.99; %0.99
r0Max = 1; %1
r0Min = 0; %0
AMax = 2; %2
AMin = 0; %0.5
freqDMax = 2; %2
freqDMin = 0; %0.1
%pa = 0.25; %0.25
wMax = 0.9; %0.9
wMin = 0.2; %0.2
xsi_init = 0.6; %0.6
```

```
n = 2; %2
% set the parameters
lb= -5 * ones(1,dim ); % Lower bounds
ub= 5 * ones(1,dim ); % Upper bounds
alfa=0.632;
beta=3/2;
sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta-1)/2)));
Vhist = zeros(10,pop,dim);
vLb = 0.6 * lb; %0.8
vUb = 0.6 * ub; %0.8
r = rand( pop, 1 ) .* 0.2 + 0;
r0 = rand( pop, 1 ) .* ( r0Max - r0Min ) + r0Min;
A = rand( pop, 1 ) .* ( AMax - AMin ) + AMin;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Initialization
for i = 1 : pop
x( i, : ) = lb + (ub - lb) .* rand( 1, dim ); %#ok<*AGROW>
v( i, : ) = rand( 1, dim );
fit( i ) = feval(FUN,x(i,:)');
end
pFit = fit; % The individual's best fitness value
pX = x; % The individual's best position corresponding to the pFit
[ fMin, bestIndex ] = min( fit ); % fMin denotes the global optimum
% bestX denotes the position corresponding to fMin
bestX = x( bestIndex, : );
Vhist(1,:,:)= v;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start the iteration.
for iteration = 1 : M
%%%%%%%%%%%%%%%%%%%%%% iterative parameters %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
freqD = rand( pop, dim ) .* ( freqDMax - freqDMin ) + freqDMin;
w = (wMax - wMin) * ( M - iteration )/(1.0 * M) + wMin; %Inertia weight
xsi1 = 1 + ((xsi_init-1)*((M-iteration)/M)^n);
xsi2 = 1-xsi1;
meanA = mean( A );
meanP = mean(pX);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 1 : pop
if rand < 0.5 %0.5
q1 = randi([1 pop]);
q2 = randi([1 pop]);
X1 = pX(q1,:);
X2 = pX(q2,:);
newX=kill_bat(pX(i,:),X1,X2);
[pX(i,:),pFit(i)]=select_bat(FUN,pX(i,:),newX,pFit(i));
%%%%%%%%%%%%%%%%%%%%%%%%%%%% L\'{e}vy flight %%%%%%%%%%%%%%%%%%%%%%%%%%%%
u=randn(1,dim)*sigma;
VV=randn(1,dim);
step=u./abs(VV).^(1/beta);
stepsize = 0.01*step.*randn(1,dim); %0.01
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x( i, : ) = pX(i,:) + stepsize .* ... %pX(i,:)
abs(meanP - x(i,:)); % change first x to pX
else
t1 = randi([1 10]);
q = randi([1 pop]);
while q== i
q = randi([1 pop]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% fractional derivative %%%%%%%%%%%%%%%%%%%
```

```
Vh = reshape(Vhist(:,i,:),10,dim);
vout = Frac_Diff_Der(alfa,Vh,t1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v(i,:) = w.* vout + freqD(i,:) .* xsi1 .*(bestX-x(i,:))+freqD(i,:).* xsi2
v(i,:) = Bounds(v(i,:),vLb,vUb);
x(i,:) = x(i,:) + v(i,:);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
% Local search
if rand > r(i)
%%%%%%%%%%%%%%%%%%%%%%% another L\'{e}vy flight %%%%%%%%%%%%%%%%%%%%%%%%
u=randn(1,dim)*sigma;
VV=randn(1,dim);
step=u./abs(VV).^(1/beta);
stepsize = 0.01*step;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
randnValueA = stepsize .*randn(1,dim).*(abs(A(i)-meanA)+realmin);
x(i,:) = bestX .* (1+randnValueA);
end
x(i,:) = Bounds(x(i,:),lb,ub);
fit(i) = feval(FUN,x(i,:)');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Update the individual's best fitness value and the global best one
for i = 1 : pop
if fit( i ) < pFit( i )
pFit( i ) = fit( i );
pX( i, : ) = x( i, : );
end
if( pFit( i ) < fMin && rand < A(i) )
```

```
fMin = pFit( i );
bestX = pX( i, : );
A(i) = A(i) * alpha;
r(i) = r0(i) * ( 1 - exp( -gama * iteration ) );
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%newx=remove_bats(pX,lb,ub,pa);
%[pX,pFit]=get_best_bats(FUN,pX,newx,pFit);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Vhist(2:10,:,:) = Vhist(1:9,:,:);
Vhist(1,:,:) = v;
fbest = fMin; %#ok<NASGU>
FunMin(iteration) = fMin;
if feval(FUN, 'fbest') < ftarget % COCO-task achieved
break; % (works also for noisy functions)
end
end
time = toc;
% End of the main program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following functions are associated with the main program
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Application of simple limits/bounds
function s = Bounds( s, Lb, Ub)
% Apply the lower bound vector
temp = s;
I = temp < Lb;
temp(I) = Lb(I);
% Apply the upper bound vector
```

```
J = temp > Ub;
temp(J) = Ub(J);
% Update this new move
s = temp;
%%%%%%%%%%%%%%%%%%%%%%%%%% from CS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [X,fits]=get_best_bats(FUN,X,newX,fits)
% % Evaluating all new solutions
%
% for j=1:size(X,1)
% fnew=feval(FUN,newX(j,:)');
% if fnew<=fits(j)
% fits(j)=fnew;
% X(j,:)=newX(j,:);
% end
% end
% function newX=remove_bats(X,Lb,Ub,pa)
% % A fraction of worse nests are discovered with a probability pa
% n=size(X,1);
% % Discovered or not -- a status vector
% K=rand(size(X))>pa;
% stepsize=rand*(X(randperm(n),:)-X(randperm(n),:));
% newX=X+stepsize.*K;
% for j=1:size(newX,1)
% s=newX(j,:);
% newX(j,:)=Bounds(s,Lb,Ub);
% end
function [X,fits]=select_bat(FUN,X,newX,fits)
% Evaluating all new solutions
fnew=feval(FUN,newX');
if fnew<=fits
```

```
fits =fnew;

X=newX;

end

function newX=kill_bat(X,X1,X2)

% A fraction of worse nests are discovered with a probability pa

stepsize=rand*(X1-X2);

newX=X+stepsize;
```

Appendix E

Background

In this Appendix, the three MAs used in the present thesis are described: ABCA, BA, and FPA.

E.1 Artificial Bee Colony Algorithm (ABCA)

ABCA is an evolutionary algorithm for global search with multi-dimensions. It is considered as one of the famous algorithms of swarm intelligence. It was defined by Karaboga (2005) [59, 40].

Actionable, it depends on the foraging behavior of honey bee swarm on finding food source "nectar". This Algorithm is categorized into three groups; employed bees, onlookers and scouts. The half of this swarm are called the employed bees that have found a food source for exploiting.

After it returns to the hive and share the information about the nectar by dancing in the dance area with other bees who is known as onlookers that are considered the other half, where they are waiting in the hive to receive that information. After finishing a food source the employed bee send one of these bees to carry out a random search, which called "Scout." The position of a food source represents a possible solution of the optimization problem and the nectar amount of a food source corresponds to the fitness of the associated solution. Now, we use ABCA to optimize the following function:

$$F(Y_i) = F(y_{1,i}, y_{2,i}, ..., y_{j,i}, ..., y_{d,i}) = \sum_{j=1}^{d} \left(\frac{y_{j,i} - y_{j-1,i}}{h} - f\left(x_{j-1,i}, y_{j-1,i}\right) \right)^2 \qquad \text{(E.1)}$$

Where $y_{j,i} \in [y_{j,\min}; y_{j,\max}], i \in \{1, 2, ..., n\}$ and $j \in \{1, 2, ..., d\}, n$ is the number of employed bees, d is the dimension of the solution space. The main steps of the algorithm are given below [59]:

Step 1: Initialize the initial swarm $Y_i = (y_{1,i}, y_{2,i}, ..., y_{j,i}, ..., y_{d,i})$ by using equation

$$y_{j,i} = y_{j,\min} + rand[0; 1] * (y_{j,\max} - y_{j,\min}) \tag{E.2}$$

Calculate $F(Y_i)$ by using equation (E.2) and the fitness (fit_i) of each food source by using equation

$$fit_i = \begin{cases} \frac{1}{1+F(Y_i)} & \text{if} \quad (F(Y_i) \geq 0) \\ 1 + abs(F(Y_i)) & \text{if} \quad (F(Y_i) < 0) \end{cases} \tag{E.3}$$

Step 2: (Move the employed bees)

Calculate the new solution $y^*_{j,i}$ by using equation

$$y^*_{j,i} = y_{j,i} + rand[-1; 1] * (y_{j,i} - y_{j,k}), \tag{E.4}$$

Where $i \neq k$ and $i, k \in \{1, 2, ..., d\}$, j, k are selected randomly and $y_{j,k}$ is a neighbor bee of $y_{j,i}$. Calculate $F(Y^*_i)$ by using equation (E.1) and its fitness (fit_i) by using equation (E.3), after that we compare this fitness with its old one. If the new food source fitness has equal or better than the old fitness, the old one is replaced by the new one. Otherwise, the old one is retained.

Step 3: (Move the onlookers)

Calculate The probability p_i of selecting the food source i by

$$p_i = \frac{fit_i}{\sum_{j=1}^{d} fit_j} \tag{E.5}$$

For improving the solution Y_i we use the main operations of Step 2.

Step 4: (Move the Scouts) If the fitness values of the employed bees are not improved by a continuous predetermined number of iterations, which is called (Limit) those food sources are abandoned, and these employed bees become the scouts, and by using equation (E.2) generate a new solution for the employed bee.

Step 5: If the termination condition is met, the stop and the best food source is memorized; otherwise the algorithm returns to Step 2.

E.2 Bat Algorithm (BA)

BA was introduced by Xin-SheYang in 2010 [88]. It simulates the echolocation behavior of micro bats. It is based on three important rules.

1. For sensing distance, bat uses its echolocation capacity. It also uses echolocation to differentiate between food and prey and background barriers even in the darkness.

2. Bats used to fly randomly with some characteristics like a velocity, fixed frequency and loudness to search for a prey.

3. It also features the variations in the loudness from a large loudness to minimum loudness.

Bats find the prey using varying wavelength and loudness while their frequency, position and velocity remains fixed. They can adjust their frequencies according to pulse emitted and pulse rate. The algorithm starts with initialization of population of bats. Each bat is assigned a starting position which is an initial solution. The pulse rate and the loudness are defined randomly. Every bat will move from local solutions to global best solutions after every iteration. The values of pulse emission and loudness are updated if a bat finds a better solution after moving. This process is continued till the termination criteria is satisfied. The solution so achieved is the final best solution. The main steps of BA are showed below:

Pseudo code of the bat algorithm (BA).

Objective function $f(x)$, $x = (x_1, ..., x_d)^T$

initialize the bat population x_i $(i = 1, 2, ..., n)$ and v_i

Define pulse frequency f_i at x_i

Initialize pulse rates r_i and the loudness A_i

while $(t < Max\ number\ of\ iterations)$

Generate new solutions by adjusting frequency,

and updating velocities and locations/solutions

    if $(rand > r_i)$

        Select a solution among the best solutions

        Generate a local solution around the selected best solution

    end if

        Generate a new solution by flying randomly

    if $(rand < A_i\ \&\ f(x_i) < f(x_*))$

        Accept the new solutions

        Increase $r_i$and reduce A

    end if

Rank the bats and find the current best x_*

end while

Postprocess results and visualization

E.3 Flower Pollination Algorithm (FPA)

Flower pollination algorithm is the latest bio-inspired algorithm proposed by Xin-She Yang in 2012 [90]. It is inspired by fertilization (pollination) process of flowers.

In FPA, abiotic and self-pollination are considered for local pollination while biotic and cross-pollination is considered for the global pollination between the flower plants. The algorithm maintains a balance between local and global pollination. Yang assumed that each plant can have only one flower and each flower can have only one pollen grain for the purpose of optimizing the benchmark functions.

The process of pollination is done by pollinators such as flies, insects or wind. Thus, each

flower (or pollen) can be considered as a potential solution of an objective function. The objective function finds the best flower, which is capable of doing maximum pollination. The process continues unless stopping criteria is met. The main steps of FPA, or simply the flower algorithm are illustrated below:

Pseudo code of the (FPA).

Objective min or max $f(x), x = (x_1, x_2, ..., x_d)$

Initialize a population of n flowers/pollen gametes with random solution

Find the best solution g_*in the initial population

Define a switch probability $p \in [0, 1]$

while (t< MaxGeneration)

  for $i = 1 : n$ (all n flowers in the population)

    if $rand < p$,

      Draw a (d-dimensional) step vector L which obeys an Lévy distribution

      Global pollination via $x_i^{t+1} = x_i^t + L(g_* - x_i^t)$

    else

      Draw ϵ from a uniform distribution in $[0, 1]$

      Randomly choose j and k among all the solutions

      Do local pollination via $x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t)$

    end if

      Evaluate new solutions

    If new solutions are better, update them in the population

  end for

Find the current best solution g_*

end while
