Design of optimization algorithms for large scale continuous problems : application on deep learning

Léo Souquet

▶ **To cite this version:**

Léo Souquet. Design of optimization algorithms for large scale continuous problems : application on deep learning. Machine Learning [cs.LG]. Université Paris-Est, 2019. English. NNT : 2019PESC0089 . tel-03477086

HAL Id: tel-03477086
https://theses.hal.science/tel-03477086v1

Submitted on 13 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Université Paris Est

Thèse de doctorat

En informatique

par

Léo Souquet

École doctorale de Mathématiques et STIC (MSTIC, E.D. 532)
Laboratoire Images, Signaux et Systèmes Intelligents (LISSI, EA-3956)

Design of optimization algorithms for large scale continuous problems: Application on Deep Learning

Soutenue le 19 décembre 2019 à Data ScienceTech Institute
devant le jury composé de:

Kalyanmoy, DEB	Professor	Michigan State University	Rapporteur
Swagatam, DAS	Associate Professor	Indian Statistical Institute	Rapporteur
El-Ghazali, TALBI	Professeur des Universités	Université Lille 1	Examinateur
Hugues, TALBOT	Professeur des Universités	CentraleSupelec	Examinateur
Nadiya, SHVAI	Scientist	VINCI	Examinateur
Amir, NAKIB	Maître de Conférences HDR	Université Paris-Est Créteil	Directeur

December 2019

Acknowledgements

I would like to thank all members of the *Laboratoire Images, Signaux et Systèmes Intelligents* and its director, Prof. Yacine Amirat.

I would like to thank everyone close to me who have supported me for the last three years. My Mother and my Father for their unconditional love and support. My Dear Blandine (and her family) for always making me smile and taking care of me. My friends for allowing me to take a step back when I could not see how to progress further.

A special thank you for those who have allowed me to pursue my work. Adel who brought me halfway across the world when I needed it. Sebastien who always pushed me through my studies and who has been a mentor and a friend. Jennifer for taking care of me every time I was in Nice. To Franck, Laurie and their beautiful family for feeding me every time I came to visit. Cédric for the countless exchanges on work and technology. Nadiya for her kind advice, support and for doing her magic. Benedicte for her trust, guidance and also for being here in difficult times.

I would also like to thank the DSTI family. Vincent who has been here from the beginning and helped me countless times. To Sarah for her support and positive energy. To Katia for her kind words. To Laurence for proof-reading part of this document. To Johnny who always makes me laugh. To Alain Bravo for his inspiration and his support. To José Massol for allowing me to start my thesis, for all his advice and most importantly for believing in me at the beginning of our project. And of course, to all my students for challenging me during my classes.

To everyone I met during workshops and conferences for sharing their own experience, helping me see through my situation.

Finally and most importantly I would like to thank my director Amir Nakib who has first accepted me as his student, guided me for three years and has always been patient with me. He has been a mentor and an incredible advisor all those years. He kept me motivated and allowed me to achieve this important step in my life. He trusted me when possible and set me straight when needed. I will always remember his advice in the difficult times and can only hope that the work we achieved together has made him proud.

PS: I would like to sincerely thank everyone who reads, even partially, this manuscript and I hope you enjoyed reading it as much as I enjoyed writing it.

Abstract

This last decade the complexity of the problems increased with the increase of the CPUs' power and the decrease of memory costs. The appearance of clouds infrastructures provide the possibility to solve large scale problems. However, most of the exact and stochastic optimization algorithms see their performances go down with the increase of the dimension of the problems. Evolutionary approaches and other bio-inspired approaches were widely used to solve large scale problems without lot of success. Indeed, the complexity of large scale problems non convex functions comes from the fact that local minima (and maxima) are rare.

In this thesis, we propose to tackle large scale problems by designing a new approach based on fractal decomposition of the search space using hyperspheres. This geometrical decomposition allows the algorithm to be intrinsically parallel for solving large scale problems. The proposed algorithm called Fractal Decomposition Algorithm (FDA). It is a deterministic algorithm with low complexity and easy to implement. FDA has been tested on several functions, compared with competing metaheuristics and showed good results on problems with dimensions from 50 to 1000. Its structure allows it to be naturally parallelized, which resulted in developing two new versions: PFDA for multi-threaded environments and MA-FDA for multi-nodes environments. Then, the proposed algorithm was adapted to solve multi-objective problems. Two algorithms were proposed: the first one is based on scalarization and has been distributed on multi-node architecture virtual environments known as containers. While the second approach is based on sorting of non-dominated solutions.

Moreover, we applied FDA to the optimization of the hyperparameters of deep learning architectures with a focus on Convolutional Neural Networks. We present an approach using bi-level optimization separating the architecture search composed of discrete parameters from hyperparameter optimization with the continuous parameters. This is motivated by the fact that automating the construction of deep neural architecture has been an important focus over recent years as doing it manually is very time consuming and prone to error.

Keywords: continuous optimization, metaheuristic, multi-objective, deep learning, fractals, large-scale optimization bi-level otpimization.

Résumé

Cette dernière décennie, la complexité des problèmes s'est accrue avec l'augmentation de la puissance des processeurs et la diminution des coûts de mémoire. L'apparition d'infrastructures *cloud* offre la possibilité de résoudre des problèmes en grandes dimensions. Cependant, la plupart des algorithmes d'optimisation exacts et stochastiques voient leurs performances diminuer avec l'augmentation de la dimension des problèmes. Les approches évolutionnaires et autres approches bio-inspirées ont été largement utilisées pour résoudre des problèmes à grande échelle sans grand succès. En effet, la complexité de ces problèmes aux fonctions non convexes vient du fait que les minima (et maxima) locaux sont rares.

Dans cette thèse, nous proposons d'aborder des problèmes à grande échelle en concevant une nouvelle approche basée sur la décomposition fractale de l'espace de recherche par hypersphères. Cette décomposition géométrique permet à l'algorithme d'être intrinsèquement parallélisable. L'algorithme proposé est appelé *Fracal Decomposition Algorithm* (FDA). Il est déterministe, de faible complexité et facile à implémenter. FDA a été testé sur plusieurs fonctions, comparé aux métaheuristiques concurrentes et a montré de bons résultats sur des problèmes de dimensions allant de 50 à 1000. Sa structure lui permet d'être naturellement parallélisée, ce qui a permis de développer deux nouvelles versions : PFDA pour les environnements multi-threaded et MA-FDA pour les environnements multi-nœuds. Ensuite, l'algorithme proposé a été adapté pour résoudre des problèmes multi-objectifs. Deux algorithmes ont été proposés : le premier est basé sur la scalarisation et a été distribué sur une architecture multi-nœuds grâce à des conteneurs. La seconde approche est basée sur le tri de solutions non dominées.

De plus, nous avons appliqué FDA à l'optimisation des hyperparamètres des architectures d'apprentissage profond en mettant l'accent sur les réseaux neuronaux convolutionnels. Nous présentons une approche utilisant l'optimisation à deux niveaux séparant la recherche d'architecture composée de paramètres discrets de l'optimisation des hyperparamètres avec les paramètres continus. Ceci est motivé par le fait que l'automatisation de la construction de l'architecture neuronale profonde a été une priorité importante ces dernières années, car le travail manuel prend beaucoup de temps et est sujet aux erreurs.

Mots Clés : optimisation continue, métaheuristique, multi-objectifs, deep learning, fractals, optimisation en grandes dimensions, optimisation bi-niveaux.

Contents

General Introduction

In the last decade, the complexity of the problems being handled could be increased due to the increase of the CPUs' power and the decrease of memory costs. Indeed, the clouds and other supercomputers provide the possibility to solve large scale problems. However, most of the exact and stochastic optimization algorithms see their performances go down with the increase of the dimension of the problems. Complexity in an optimization problem can be due to the non-linearities, multi-modality, computational time to evaluate the cost function or even uncertainties in parameters. In this thesis, we decided to address the dimensional complexity.

This thesis addresses those optimization problems. Different techniques have been developed in the literature, among them heuristics and metaheuristics. Indeed, metaheuristics address the complexity and variety of problems by being designed to solve hard optimization problems, without any knowledge about the considered problem. However, their stochastic nature is a limiting factor, in some applications, when it comes to safety critical applications where repeatability is important. Typically in these cases, metaheuristics can be used to improve the parameter settings of deterministic algorithms. Moreover, when the metaheuristics are efficient, some of them are difficult to implement. Beside, the justification of an obtained solution can also be difficult because the method used is based on a complex stochastic search rather than on a deterministic approach.

Furthermore, the complexity of large scale problems comes from the fact that local minima (and maxima) are rare compared to saddle points. Indeed, some points around a saddle point have greater fitness (value of the objective function) than the saddle point, while others have a lower fitness value. This phenomena can be explained by the fact that at a saddle point, the Hessian matrix has both positive and negative eigenvalues. Then, points lying along eigenvectors associated with positive eigenvalues have greater fitness than the saddle point, while points lying along negative eigenvalues have a lower value. Consequently, a saddle point can be considered as a local minimum along one cross-section of the fitness function and a local maximum along another cross-section

[Nakib et al., 2017]. For instance, let $f : R^n \rightarrow R$ be a function of this type, the expected ratio of the number of saddle points to local optima grows exponentially with n. To understand the perception following this behavior, one can see that the Hessian matrix at a local minimum has only positive eigenvalues, but the Hessian matrix at a saddle point has both positive and negative eigenvalues. Assume that the sign of each eigenvalue is generated by flipping a coin, then, in one dimension, it is easy to obtain a local minimum by flipping a coin and getting heads once. In n-dimension case, it is exponentially unlikely that all n coins tossed will have heads outcome. To solve large-scale problems, evolution algorithms, evolution strategy, and differential evolution algorithms have been extensively modified and adapted for different problems. Those modifications do increase the performances but also increase significantly the complexity of implementation.

The goal of this work is to design an efficient optimization algorithm to deal with large scale optimization problems. The idea is to propose a low complex approach and easy to implement. Then, the proposed algorithm is called, "Fractal Decomposition based Algorithm" (FDA). It is based on a fractal geometrical decomposition of the search space. The main principle of the approach consists of dividing the feasible search space into sub-regions with the same geometrical pattern. Indeed, decomposing the search space allows the algorithm to create a tree composed of sub-regions. This principle makes it naturally parallelizable. The idea is to benefit from modern infrastructure and allow FDA to run on multi-threaded and muti-node environments.

This thesis has been funded by Data ScienceTech Institute and prepared within the University Paris-Est Créteil (UPEC), in the *Laboratoire Images, Signaux et Systèmes Intelligents* (LiSSi, E.A. 3956) under the direction of the Amir Nakib (PhD) within the group SIMO (Signal, Image and Optimization).

The main contribution of this thesis are:

- The design of a new deterministic metaheuristics, called FDA, based on a fractal geometrical decomposition of the search space using hyperspheres, capable of solving large-scale black-box problems.
- The adaptation of our algorithm to benefit from modern distributed architectures.
- Two new approaches for solving Multi-Objective problems. The first one, Mo-FDA-S uses scalarization and the other, Mo-FDA-D leverages non-dominated sorting.

- The application of FDA to design and optimize Convolutional Neural Networks (CNN) using bi-level optimization and the distinction between discrete and continuous parameters.

This manuscript is organized as follow:

The first chapter introduce the state-of-the art on metaheuristics, followed by the literature on parallel metaheuristics and multi-objective approaches.

The second chapter presents the proposed algorithm called "Fractal Decomposition based Algorithm" (FDA). The use of hyperspheres as an elementary geometric form is considered. This choice was motivated by its low complexity and flexibility to cover a part of the search space. Indeed, it is easy to go from one center of a hypersphere to another analytically without storing them. A performance analysis is conducted on large-scale global optimization test functions with dimensions going from 50 to 1000. The obtained results and the comparison with other metaheuristics designed to solve the same types of problems, show the efficiency and the competitiveness of the proposed algorithm to solve large scale optimization problems.

In its original version, FDA was running on a mono-threaded environment and therefore its computational time increases significantly when the problems' dimension increases. In this chapter we tackle this problem by proposing two different solutions. Reducing the execution time while maintaining the original precisions. The first approach benefits from multi-threaded environments while the other from multi-node environments. Both approaches are designed to leverage current available Information Technology resources such as new cloud infrastructures. The multi-threaded FDA is called, "Parallel Fractal Decomposition based Algorithm" (PFDA) [Nakib et al., 2018] and the multi-node is called "Multi-Agents Fractal decomposition based algorithm" (MA-FDA).

The fourth chapter presents the adaptation to tackle multi-objective problems. In multi-objective optimization problems (MOP) the goal is to optimize at least two objective functions simultaneously. In this chapter, two new approaches have been developed. Mo-FDA-S based on the scalarization approach using the Tchebycheff technique to decompose the objective space. This approach has also been developed to benefit from a multi-node environment to improve the computational time. This architecture takes profit from containers, lightweight virtual machines that are designed to run a specific task only. The second approach, Mo-FDA-D uses the principle of non-dominated sorting.

The fifth chapter presents an application of FDA to the optimization of the hyperparameters of deep neural network architectures. We present an approach using bi-level optimization separating the architecture search composed of discrete parameters from

hyperparameters optimization with continuous parameters. This is motivated by the fact that automating the construction of deep neural architecture has been an important focus over recent years as doing manually is very time consuming and prone to error and that the experts doing that are expensive in the industry.

Finally, a general conclusion summarizes our results and contributions as well as presenting the potential future work.

Chapter 1

State-of-the Art on metaheuristics

1.1 Introduction

Optimization is at the crossroad of mathematics and computer science. It is the study of problems in which we wish to find the best solution among a set of feasible ones. Problems can be divided into three categories. The first one is *discrete optimization* which refers to problems with discrete variables. The second, *continuous optimization* refers to continuous variables as well as a continuous function being optimized. The third, called "Mixed-interger programming" methods, are common in practice, and are used when the set of decision variables contains integers and real values. In this thesis, we focus on dimensional complexity.

The term black-box refers to the fact that the algorithm has no information on the function being optimized and does not (usually) assume any analytical form. Mathematically, an unconstrained optimization problem with box constraints is represented as follows:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{X}, \mathcal{X} = \left\{x \in \mathbb{R}^D : l \leq x \leq u\right\} \end{array} \tag{1.1}$$

where $f(x)$ is the function to optimize and is called the objective function or cost function. $\mathcal{X}$ is the set of feasible solutions in the search space, bounded by the lower bound l and upper bound u and D refers to the dimension of the problem, *i.e.* the number of decision variables. Optimization aims to find the extremum (minimum or maximum). The global optimum, x_0, in case of minimization is defined as:

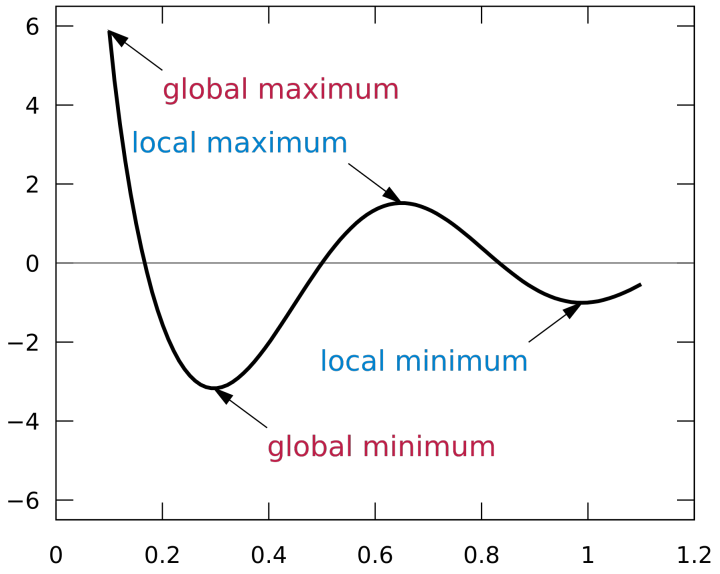


FIGURE 1.1: Illustration of the different extremum of function.

$$x_0 \in \mathrm{X} \text{ is a global minimum point of function } f : \mathrm{X} \to \mathbb{R} \text{ if } (\forall x \in \mathrm{X}) f(x_0) \leq f(x) \tag{1.2}$$

By convention, in optimization, the standard form defines a minimization problem. A maximization problem can be treated by negating the objective function.

Global optimality and local optimality are two important notions in the field of optimization. The former is defined in Equation 1.2 and the intuition is that it is the best solution(s) that can be found for a given problem. However, one of the issue when optimizing a function is a local optimum. In X, x_{local} is a local optimum if there exists some $\epsilon > 0$ such that $f(x_{local}) \leq f(x)$ for all $x \in X$ where $||x - x_{local}|| \leq \epsilon$. The intuition here is that x_{local} is the best solution in its neighbourhood and can be confused with the global optimum. Figure 1.1 illustrates the different extremum.

To address those optimization problems, different techniques have been developed, among them heuristics and metaheuristic are popular. According to the authors in [Clautiaux et al., 2004], heuristics returns good results in short computation time, whereas the metaheuristic aims to returns the best results known, *i.e.* the global optimum of the problems. However, other elements found in the literature can be added. Heuristics tend to be problem-dependent designed or modified to solve one problem in particular whereas metaheuristics are problem agnostic and are designed to solve any given problems without knowing the analytical form of the function being solved. The terms

metaheuristic also refers to approaches being able to transcend local optimality using different strategies [Glover, 1986].

When talking about metaheuristic, two notions must be taken into account. Diversification and Intensification are two main mechanisms that every solution in trying to balance. The former designates the fact that the algorithm will look for promising regions within the search space while it will try to find a better solution within those "good" regions during the intensification phase.

The focus here will be on metaheuristics as the work presented in this thesis concerns the development of a new metaheuristic. Many categories can be found in the literature such as single-solution based algorithms, population-based approaches, metaheuristics for multi-objective optimization, hybrid or even parallel algorithms [Talbi, 2009]. The Figure 1.2 shows a classification of the main optimization methods. In red are highlighted the main categories presented in this thesis.

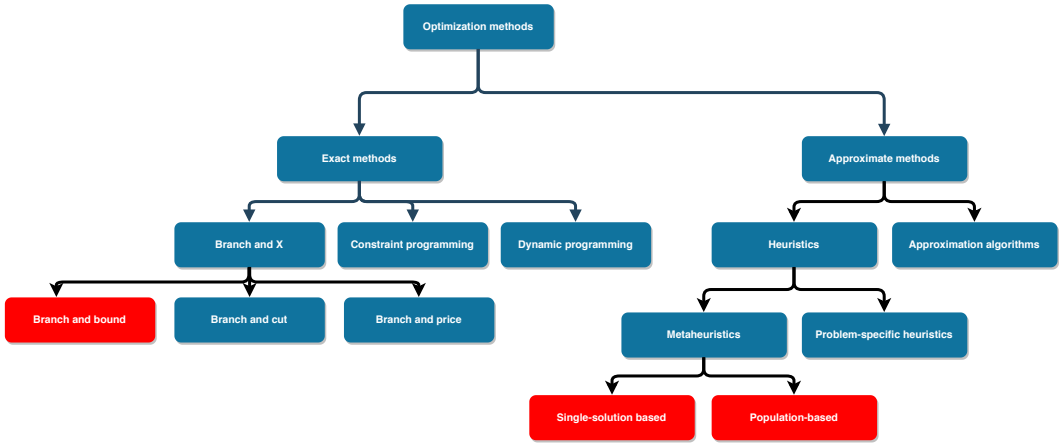


FIGURE 1.2: Classification of optimization methods [Talbi, 2009]

This chapter will first present the well-known single-solution based metaheuristics in Section 1.2. We will then cover Population-Based approaches in Section 1.3 followed by the performance assessment of metaheuristics in Section 1.4. Decomposition-based approaches are detailed in Section 1.5. Parallel algorithms will be covered in Section 1.6 as well as Multi-Objective optimization techniques in Section 1.7. This chapter will be concluded in Section 1.8.

1.2 Single-Solution metaheuristics

This category is dedicated to algorithms using one solution when solving a given problem. They focus on iteratively improving one candidate solution that "walks" through a search trajectory or neighbourhoods within the search space. The notion of neighbourhoods

is crucial in single optimization as it defines the feasible solutions around the main one being improved. Metaheuristics will implement different technics to select the next solution within the neighbourhood (or not in certain cases). This might lead algorithms to remain in a local area and miss the global optimum. To avoid local optimality, metaheuristics have implemented different strategies.

1.2.1 Local Search

Local search is probably the oldest and simplest method [Talbi, 2009] and has been designed to solve hard optimization problems in a reasonable time. The aim is not to find the global optimum but to provide a good local optimum. The principle of a local search is to navigates iteratively through a neighbourhood. Then at each step, a solution within the surrounding candidates that improves the current solution is selected. The algorithm will stop when the stopping criterion has been reached (computation time or when no candidates around in the neighbour provide an improvement of the objective function). It is important to mention that this search is not done using any gradient methods as it may seem. The search is done by generating a set of candidates around the current solution and selecting the one that improves an objective function. A template of a local search can be seen at the figure Algorithm 1. When selecting the best candidates solutions different strategies can be used. Within a neighborhood, either the solution that improves the best the function evaluation or the first solution encountered that improves it.

Algorithm 1: Generic template of a local search (in case of minimization)

Generate an initial solution s_0
Generate a current solution s and assign it s_0: $s = s_0$
Initialize the best current solution $bestSol = f(s)$
while *Stopping criterion is not reached* **do**
  Generate a set X of k candidate solutions around s
  if $f(s_i) < f(s)$ *for* $s_i \in X$ **then**
    $bestSol = f(s_i)$
    $s = s_i$ for $s \in X$
  end
  else
    Stop the local search
  end
end
Output: bestSol

Local searches can be efficient methods but the main issue is that remain stuck within local optimum. This is why more advanced method to avoid this local optimality have been developed in the literature.

1.2.2 Simulated Annealing

This metaheuristic was developed by Kirkpatrick et al. [1983]. Authors illustrated that the work done in the later to approximate numerical simulation of the behavior of a many-body system at finite temperature is suitable to be adapted as an optimization algorithm. In other words, it is inspired by the work done in metallurgy where a material is heated at an important temperature and then slowly cooled by a controlled technique aiming to obtain a strong crystalline structure. The strength obtained by the cooling system depends on the initial temperature and the rate of cooling. If the temperature is not high enough or the cooling too fast, the structure obtained will not be optimal because strong crystals are grown controlled slow cooling. This physic principle is taken from the work done in Metropolis et al. [1953].

The algorithm starts from an initial solution with a high-temperature T. At each iteration, a random solution is generated. Candidates are always accepted if it improves the cost function. If not, the candidates are still accepted with a certain probability that depends on the current temperature and the difference in value between the two solutions (current and candidate). As the algorithm progress, similar to the cooling system, the temperature decreases, hence decreasing the probability to accept the worst candidates. The probability follows the *Boltzmann distribution* as per Equation 1.4.

$$P = \exp\left(\frac{-\Delta E}{T}\right) \tag{1.3}$$

$$with\ \Delta E = f(s^{'}) - f(s) \tag{1.4}$$

where s is the current solution, $s^{'}$ the candidate solution and ΔE the difference between them. If $\Delta E \leq 0$, then $s^{'}$ is always accepted, however if $\Delta E > 0$, $s^{'}$ is accepted with probability P. At the beginning of the algorithm, T is high, therefore, non-improving solutions are likely to be accepted. However, as the algorithm progress, T is reduced and so is the P. Simulated Annealing (SA) has been originally designed to solve discrete optimization problems however in the literature it has been modified to solve continuous problems [Siarry et al., 1997]. SA has been modified in different studies to address large-scale optimization problems. In Hasançebi et al. [2010] authors mentioned that SA fails to produce acceptable solutions to high dimensional problems due to its poor convergence characteristics. Their solution did enhance SA performances but added two more parameters which increase its complexity.

In addition to the issues common to all metaheuristics, the specific issues with Simulated Annealing are both the acceptance probability being the main element in the algorithm

and the cooling schedule, *i.e.* at which step is the Temperature T decreased and by how much. In his paper, F. Glover [Glover, 1986] argued about the random selection implemented in simulated annealing and proposed a controlled selection in his algorithm called "Tabu Search".

1.2.3 Tabu Search

Tabu search is a metaheuristic that has been developed by F. Glover in his famous paper *Future paths for integer programming and links to artificial intelligence* [Glover, 1986]. He pointed out the stochastic behavior of Simulated Annealing and proposed a deterministic algorithm. Tabu searched was designed under the supposition that there is no value in choosing a non-improving candidate except if it is to avoid taking a path already examined. Tabu Search can be seen as a local search. Iteratively, the algorithm will move from one solution to another, choosing the best neighbour at each iteration. To avoid local optimality, if no candidates solutions improve the current one, the best among them will be chosen. However, the significant improvement brought by this algorithm lies in the construction of a memory list while exploring the search space. A list of the m most recent moves are recorded in the order in which they have been made and added at the end of the list. This list contains the "tabu" solutions that have been visited already and ensures that the algorithm does not visit the same path twice. The balance between diversification and intensification is done by tuning the size of the list m. It is an important parameter. The larger it is, the more diversified the algorithm is, visiting a larger part of the search space as many moves will be restricted. However, if m is short, the Tabu Search behaves as having a "goldfish memory" and will forget previous moves and will tend to intensify more.

The issues of the original Tabu Search lies in the used of only one list which can be restrictive and in the fine-tuning of m. However, many modifications have been made in the literature to improve the algorithm. In [Glover, 1997], F. Glover integrates the notion of adaptative memory. In [Chelouah & Siarry, 1999], the authors have enhanced Tabu Search for the global optimization of multi-minima functions. The algorithm has been modified to solve multimodal functions with continuous variables in [Hajji et al., 2004].

To solve high dimensional problems, Tabu Search has been modified to explore the neighborhood of the current solution gradually through smaller number of variables [Hedar & Fouad, 2012]. Indeed, authors showed that on high dimensions, exploring the neighborhood of all variables at the same time can negatively affect the progress of the search. This version shows promising results comparing to other Scatter Search methods

Algorithm 2: Generic template of the Tabu Search search (in case of minimization)

Generate an initial solution s_0
Generate a current solution s and assign it s_0: $s = s_0$
Initialise the best current solution $bestSol = f(s)$
Create an empty *tabuSearch* list and push s at the end
while *Stopping criterion is not reached* **do**
    Generate a set X of k candidate solutions around s
    Find the best candidate $s_{best} \in X$
    if $f(s_{best}) < f(s)$ *and* $s_{best} \notin tabuSearch$ **then**
        $bestSol = f(s_{best})$
        $s = s_{best}$
    end
    else if $f(s_{best}) \geq f(s)$ **then**
        $bestSol = f(s_{best})$
        $s = s_{best}$
        Push s_{best} in *tabuSearch* at the end of the list
    end
    if *Size of tabuSearch* $> m$ **then**
        Remove first element of *tabuSearch*
    end
end
Output: bestSol

and less computational expensive but is outperforms by more advanced methods such as Differential Evolution.

As mentioned, metaheuristics continuously balance the trade-off between diversification and intensification. While single-solution metaheuristics have implemented different strategies to maintain diversification. Another category of metaheuristic, population-based approaches, benefit from a population of solutions (as indicated in the name of the category) which helps to maintain the diversity.

1.3 Population-Based metaheuristics

This section covers the main well-known population-based metaheuristics. This category of algorithms maintains and improve iteratively multiple candidate solutions (a population). Many of those metaheuristics are nature-inspired algorithms. This means that the structure has been inspired by natures' behavior [Ser et al., 2019].

Population-based metaheuristics share certain common behaviour. They all start from an initial population of solutions and iteratively generate new populations. A selection phase is carried out to keep a subset of the population and these two phases are repeated until a stopping criterion is reached.

1.3.1 Evolutionary algorithms

Jean-Baptiste Lamarck (1744–1829) was the first to theorise the transmutation of species in [Lamarck, 1830] and believed life forms were created continuously by spontaneous generation. It is in 1859 that Charles Darwin publishes his famous book *On the Origin of Species* [Darwin, 1859] about his theories of evolution and natural selection. In a nutshell, it is the principle that mutation occurs from a generation to another and the beneficial ones are preserved because they aid survival. In the mid-1960s, computer scientists have seen in the theory of evolution a way to apply to optimization. Historically two main different schools can be identified. Genetic Programming in [Fogel et al., 1966] and Genetic Algorithm (GA) in [Holland, 1975], all part of Evolutionary Algorithm (EA). However other models of evolutionary algorithms have been proposed in the literature such as Differential Evolution. They are all stochastic population-based metaheuristics and iterate over many populations simulating the evolution of species. The initial population is generated randomly and at each iteration, a new one is generated. Each individual is evaluated against the cost function which indicates its relevance to the problem. The best individual is selected for "reproduction" and becomes parents. Through different strategies, new individuals are generated (children). The higher the fitness is for an individual the higher the probability it will be selected. This process is repeated iteratively until the stopping criterion is reached such as the number of function evaluations or generation. Template of an EA algorithm is shown in Algorithm 3.

Algorithm 3: Generic template of an EA algorithm (in case of minimization)

Generate an initial population P_0 randomly of n individuals
$t = 0$
while *Stopping criterion is not reached* **do**
  Evaluate each individual of population P_t
  $P_t^{'}$ = Select the k best individuals in P_t
  $P_{children}$ = Generate new individuals from $P_t^{'}$
  P_{t+1} = best individuals among $P_{children}$ and $P_t^{'}$
  $t = t + 1$
end
Output: bestSol individual or best population

Genetic algorithms were developed by J. Holland (University of Michigan, USA) in [Holland, 1975] to understand the adaptative process of evolution of a population. They are a very popular type of EA algorithm and have been widely applied to real optimization problems as well as machine learning. Originally developed for discrete optimization they have been modified to all types of optimization problems [Chelouah & Siarry, 2000]. The particularity of GA is that the generation of a new individual from a population is done using two strategies. The crossover, combines multiple parents to generate a new

one and the mutation, *i.e.* changing slightly one parent to create a new solution. After those two phases, we reevaluate all individuals and create a new generation. Both occurs with a probability $P_c \in [0, 1]$. Moreover, the offspring population always replace the parent population using different selection strategies such as proportionate, tournament or ranking [M.A. AL-Salami, 2009]. In the literature, many different operators can be used for the selection, crossover or mutation phases [Bäck et al., 2000].

1.3.2 Evolution Strategies

Evolution Strategies (ES) is another branch of Evolutionary Algorithm first proposed by Ingo Rencherberg in [Rechenberg, 1965]. This type of algorithms has been mostly applied to continuous optimization and emphasise on using normally distributed mutation. Where in GA the size of the parents and offsprings populations are similar, in Evolution Strategies, their size can be different from one iteration to another. The approach called CMA-ES *(Covariance Matrix Adaptation Evolution Strategy)* is probably the most famous ES algorithm [Hansen & Ostermeier, 2001]. It is stochastic and derivate-free, considered as one of the most powerful stochastic optimizers but the original formulation does not scale well on large optimization problems. It has been the subject of many studies and modifications in the literature [Varelas et al., 2018].

1.3.3 Differential Evolution

One of the most popular and most performant EA for solving continuous optimization problems is called Differential Evolution algorithm (DE) [Storn & Price, 1995]. Unlike the other families of EAs presented above, DE perturbs the current-generation population members with the scaled differences of randomly selected and distinct population members. Therefore, no separate probability distribution has to be used for generating the offspring. Different variants of DE have been suggested by Price et al. [2005] and are conventionally named DE/x/y/z, where x represents a string that denotes the base vector, *i.e.* the vector being perturbed, whether it is *"rand"* (a randomly selected population vector) or *best* (the best vector in the population with respect to fitness value), y is the number of difference vectors considered for perturbation of the base vector x and z denotes the crossover scheme, which may be binomial or exponential. The DE/rand/1/bi n-variant is also known as the classical version of DE. Recently, it has gained much popularity in different kinds of applications, because of its simplicity and robustness in comparison with other evolutionary algorithms [Vesterstrom & Thomsen, 2004]. DE has very few parameters to adjust, making it particularly easy to implement for a diverse set of optimization problems [A.Bastürk & E.Günay, 2009; Chang, 2006;

Yang et al., 2007]. DE has also been adapted to solve large-scale problems. A version called *Self-adaptive Differential Evolution* (SaDE) [Qin & Suganthan, 2005] uses a learning procedure to generate trial vectors strategies with their associated parameters. Another self-adaptative version called *jDElscop* [Brest & Maučec, 2011] has proven more efficient than others due to a reduced population size and a mechanism for changing the sign of the F control parameter. A survey on Differential Evolution and future research issues are presented on [Das et al., 2016].

1.4 Performance assessments

In this section, we briefly review the different methods to measure the performance of metaheuristics. As they do not guarantee to find the optimum, metrics have been defined to evaluate the quality of the solutions found, the computational time required to solve a problem as well as the robustness of an algorithm.

1.4.1 Quality of solution

The quality of the solution refers to its cost, also called fitness, obtained when evaluating the objective function. It corresponds to a numerical value quality which indicates the extent to which the solution obtained is satisfactory.

Concerning problems when the global optimum is known, the quality is defined as the distance between a solution found and the known global optimum. The smaller this distance, the better the quality. The most common expression to defined the quality is as follows:

$$|f(s) - f(s^*)| \tag{1.5}$$

where s is a solution found by the metaheuristic and s^* is the global optimum.

However, when the global optimum is not known, different methods can be used to define the quality of a solution. The best solution found by the algorithm can be used as a reference point and will be updated if a better solution is found. Others approaches define the quality according to a lower bound computed using relaxation techniques.

1.4.2 Computational effort

For any optimization algorithm, the computational analysis can be done both theoretically and empirically. The first one refers to the study of the worst-case complexity of the algorithm. Two types of complexity exist: asymptotic and average-case. In general, asymptotic complexity is not sufficient to define metaheuristic performance. The average-case complexity may, therefore, be more representative in cases where the distribution of input instances is known *a priori*.

Empirical performance evaluation can be done using different measures related to the computational time. Either the time of the CPU (central process unit) can be used, the GPU time or the internal clock of a computer. The main issue is that those metrics are dependant on characteristics of the physical machine running the algorithm such as CPU model, amount of RAM (random access memory), the operating system itself and also the programming language chosen to implement the algorithm.

To overcome this issue the number of function evaluation used to solve the problem can be used and is independent of the physical machine. However, this metric has its limitations when it comes to problems with a small number of evaluations or when they are not constant in time. Several stopping criteria can be employed such as the number of iterations or the time required to obtain a given solution

1.4.3 Robustness

To assess the robustness of a metaheuristic, one should study the variation of its results with respect to its parameters. The less the results change when the parameters change, the more robust the algorithm is. Also, applying a metaheuristic to different problems such as separable or non-separable. All of the results obtained are considered to define the robustness of the algorithm.

1.5 Decomposition-Based metaheuristics

So far the most commonly known metaheuristics have been proposed based on different strategies, inspired by physics, natures, swarm behaviours. In the section, we present another class of algorithms called “Decomposition-Based metaheuristic”. These methods are based on the strategy that consists in dividing the search space by carrying out a hierarchical partitioning. They are often referred to as “divide-and-conquer” approaches. They generate iteratively a tree composed of nodes representing subregions of the decision space with the root corresponding to the entire search space. This creates

a set of partitions over multiplied scales (nodes of a given depth corresponds to a partition of a specific scale). This family of the algorithm has been studied in the literature and are also referred as Multi-Scale Optimization (MSO) algorithms [Al-Dujaili et al., 2016b]. In our work, we were focused on this optimization approach.

1.5.1 Continuous Branch and Bound

Branch and bound is not a metaheuristic but belong to the class of exact methods. It is important to briefly mention this approach as it is a decomposition-based algorithm that has inspired different decomposition methods. Branch and bound ($B\&B$) was first proposed by Land & Doig [1960] to solve discrete optimization problems.

In addition, many deterministic optimization algorithms use a continuous branch and bound paradigm [Tuy & Horst, 1996] and many metaheuristics hybridized with branch and bound can be found in the literature [Blum et al., 2008]. A template of a branch and bound algorithm is shown in Algorithm 4. Here each node is associated with a set, characterized as a n-dimensional interval referred to as a *box*. The method starts at the root node with a set that contains all feasible solutions and explores the search space while dynamically building a tree of subregions. Interactively the algorithm visits nodes and bounds the optimal solution current visited node. If the lower bound is greater than the current best solution, the node is discarded. Otherwise, the algorithm looks for a better solution within the node. If found the solution is store and the node is further decomposed into two new subregions. A new global lower bound is set, *i.e.* the minimum lower bound of all remaining boxes. The search terminates when there is no part of the solution space to explore, and the optimal solution is the best solution found during the search. Branch and bound algorithm rely on two main mechanisms. The node selection which decides how to navigate in the generated tree and the partition mechanism which decompose regions which are not discarded. More strategies can be found in Tuy & Horst [1996]

Branch and bound algorithm does not scale well on high dimensional problems. This is due to the fact that this method has a worst-case running time as high as an exhaustive search of all potential nodes which is exponential. In short adding one decision variable could double the time required to optimize a problem. Domain reduction methods could be used to address this scaling issue.

Algorithm 4: Template of a Branch and bound algorithm

Lower Bound, $LB = -\infty$;
$f^* = +\infty$
k=1
$\mathcal{N} = \{X_0\}$
while *NotConverged(LB, f^*) and* $|\mathcal{A}| > 0$ **do**
  $X_k = SubRegionSelection(\mathcal{A})$;
  $\mathcal{A} = \mathcal{A} \backslash \{X_k\}$
  $LB(X_k) = ComputeLowerBound(X_k)$
  if $LB(X_k) < f^*$ **then**
    Upper Bound, (UB_k, x_k) = FindFeasibleSolution(X_k)
    if $UB_k < f^*$ **then**
      $f* = UB_k$
      $x^* = x_k$
      $\mathcal{A} = \{X \in \mathcal{A} : \text{NotConverged}(LB(X), f^*)\}$
    end
    $(X', X'') = SubregionDecomposition(X_k)$
    LB(X') = LB(X_k)
    LB(X") = LB(X_k)
    $\mathcal{A} = \mathcal{A} \cup \{X', X''\}$
  end
  LB = $min_{X \in \mathcal{A}} LB(X)$
  k = k+1
end
Output: (f^*, x^*)

1.5.1.1 Bounding methods

Being able to obtain information about a subregion is a key element in a deterministic decomposition algorithm. Three elements are essentials for an effective bounding method. 1) initial estimation of a subregion by the method; 2) rate of the resulting conversation bound and 3) efficiency in terms of computational time is a concern to any numerical method.

Two approaches can be used to bound a subregion. The first and most popular way is to directly compute a lower bound on the range of a given function. This range of methods belong to the family of interval analysis [Alefeld & Mayer, 2000; Moore & Bierbaum, 1979; Neumaier, 1991]. The arithmetic principle of the interval in the real number system can be extended and can be used to estimate conservatively an interval within the optimization problem referred to as the subregion of the function. Popular methods include natural interval extensions and centred forms.

Another approach is to set up a convex optimization that is guaranteed to return a lower bound of a given subregion. It considers the problem differently by relaxing in Equation 1.1, the feasible set and/or replacing the cost function with one that takes a smaller

value for each point in its space. This relaxation can be a linear or a convex program allowing the algorithm to solve the problem optimally [Bertsekas, 1995; Bertsimas & Tsitsiklis, 1998; Boyd & Vandenberghe, 2004].

1.5.1.2 Subregions selection methods

Subregions selection methods reduce the size of the considered subregion using the information about the current best fitness found. They aim to select the subregions to be discarded during the exploration of the search space. Either the discarded subregions contains no feasible solutions or no solution within it can represent an improvement. It has been shown that Branch and bound algorithms converge without it but their significantly reduce the run time. In [Ratz & Csendes, 1995], authors studied the influence of different interval subdivision selection methods on the convergence of branch and bound algorithms. They conclude that using the selection rule adapted to the problem being optimized, a significant amount of computation time can be saved. Other methods can be found in the literature such as duality-based [Tawarmalani & Sahinidis, 2004], feasibility-based [Hansen et al., 1991] or optimality-based [Ryoo & Sahinidis, 1995].

Authors in [Araya & Reyes, 2015] detailed the principle of interval branch and bound algorithms to solve global optimization problems. Two methods are presented. First, Newton-based for constrained problems and small regions to decompose. Then Relaxation-based methods are not restricted to constraints problems however, linear relaxation might lead to information loss of the original cost function.

Outside branch and bound methods, other algorithms have been developed to decompose the search space. For instance, the algorithm proposed in [Demirhan & Özdamar, 2003] partitioned the domain space into non-overlapping subregions. Their method called Fuzzy Adaptative Partitioning Algorithm (FAPA) uses a fuzzy assessment measure using information from randomly selected points. FAPA shows interesting results in test functions up to 10 variables but has not been tested on large scale problems.

Their algorithms use a geometrical form to decompose the search space. For instance, FRACTOP Demirhan et al. [1999], MOSS [Ashlock & Schonfeld, 2007] or DIRECT [Jones et al., 1993] for the most popular one are next presented as they are closely related to the work presented in this thesis.

1.5.2 FRACTOP

In 1999, Demirhan introduced a metaheuristic for global optimization based on geometric partitioning of the search space called FRACTOP Demirhan et al. [1999]. As a

decomposition-based algorithm, a tree of subregions is generated by bisecting the lower and upper bounds of every variable constituting the surface faced at the first level of the search tree. The geometrical form used in the proposed method is the hypercube. Indeed, the closure of the feasible region is divided into 2^n subregions with n corresponding to the problem dimension. Then, several solutions are collected randomly from each subregion or using a metaheuristic such as simulated annealing [Kirkpatrick, 1984] or genetic algorithm [Goldberg, 1989]. After that, a guidance system is set through fuzzy measures to lead the search to the promising subregions simultaneously and discard the other regions from partitioning. The belief property that a subregion contains the global optimum is estimated using the belief measure of the previous level and the evidence of the sample of solutions collected from each subregion of the current level. The main advantage of the decomposition procedure used in FRACTOP is that there is no overlap, avoiding to visit the same local area more than once. Thus, that makes the proposed approach efficient for low dimensions. However, the decomposition procedure generates 2^n subregions. Hence, when n is higher, the complexity of the partitioning method increases exponentially. For instance, this algorithm must visit 2^{50} subregions when solving a problem of dimension 50.

1.5.3 Multiple Optima Sierpinski Searcher

In the literature, an algorithm using a representation based on the fractal geometry for evolutionary algorithms called Multiple Optima Sierpinski Searcher was proposed in [Ashlock & Schonfeld, 2007]. Herein, the fractal geometrical form chosen for this method is the Sierpinski triangle (Figure 1.3) generated using the chaos game which consists of moving a point repeatedly from a vertex to another selected randomly. Besides, to reduce the computational cost, the located optima are stored and manipulated using strings of characters that specify them. The author proposed to face this limit to select $n + 1$ generators instead of 2^n generator samples. Authors mentioned that the chosen geometric form does not allow to cover the full search space. This was only a primary work and was never extended because of the complexity of this approach.

1.5.4 DIRECT Algorithm

The algorithm called DIviding RECTangles (DIRECT) is probably the most popular decomposition-based method in the literature. First introduced in [Jones et al., 1993] the algorithm has received great success for solving optimal design problems. It was motivated by a modification of the Lipschitzian optimization. The idea of DIRECT was to use the *Lipschitzian optimization* without the *Lipschitzian constant*. It was inspired

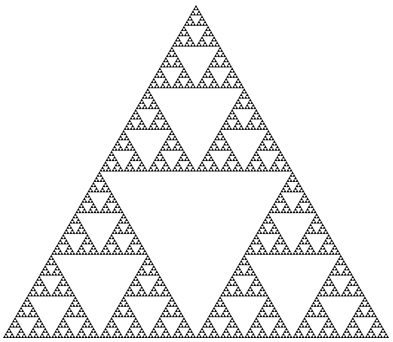


FIGURE 1.3: The Sierpinski triangle generated by a 3-cornered chaos game.

by this mathematical property but DIRECT does not require any analytical knowledge of the function f being optimized. The original approach uses hyper-rectangles to decompose the search space. It partitions the feasible search space into a growing number of hyper-intervals. Then, at each iteration, the most promising ones are selected for further partitioning. An illustration of the principle of DIRECT is shown on Figure 1.4.

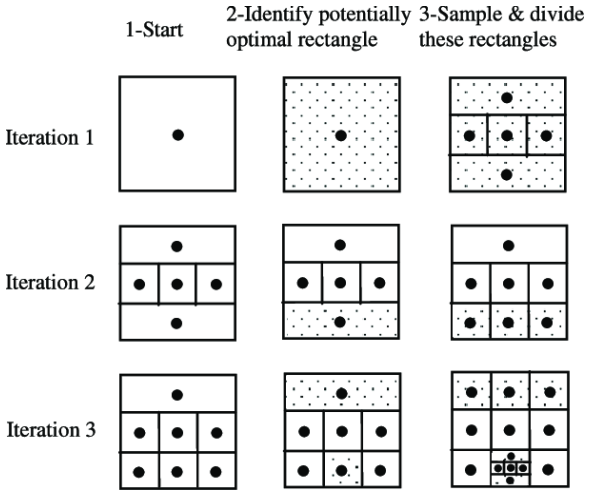


FIGURE 1.4: An illustration of the first three iterations of the DIRECT algorithm (Figure taken from Lang et al. [2007]).

As mentioned, DIRECT uses hyper-rectangles as a geometrical form to decompose the search space, moreover to select potentially optimal subregions, the distance from the center of the hyper-interval to its vertices is computed. Then, the number of vertices increases exponentially as the problem dimension increases. Consequently, the algorithm's performances to decrease drastically, in terms of computation time and quality of the final solution.

1.5.4.1 Other versions of DIRECT

As mentioned DIRECT is a popular algorithm and therefore has been the subject of many studies and improvement to overcome the main issues of the original version.

For instance, in [Gablonsky & Kelley, 2001], the authors proposed a modified version of DIRECT that is strongly biased toward local search. Instead of $L^2 - norm$ to compute the global score, they used a $L^\infty - norm$, hence reducing the variations in the global score. Even with interesting numerical results, this approach is only designed for low-dimensional problems with only a few local minima.

In [Finkel & Kelley, 2006], the authors show that the convergence behavior of (DIRECT) is sensitive to additive scaling of the cost. They illustrate this issue and proposed a new version to deal with the sensitivity. To do so they have updated the way the algorithm evaluates potential optimal hyper-rectangles. After each iteration, the solutions found are scaled by subtracting the median of the collected solutions found so far.

Another modification of DIRECT was proposed in [Liu & Cheng, 2014] to overcome the issue mentioned by Jones D. R. in [Jones et al., 1993]: while the algorithm quickly can get close to the subregion containing the global optimum, high degree of accuracy requires a high computational time. A bilevel strategy is introduced to overcome this issue. At each iteration, the modified version *RDIRECT-b* creates two levels of search spaces, one is the fine level search space, another is the coarse level search space. The former contains the whole set of hyperrectangles while the second smaller hyper-rectangles defined from an earlier interaction.

In [Paulavičius et al., 2014], the authors introduced a two-phase approach. During the first phase, the algorithm tries to explore better the subregion around the current best point. The phase ends when the improvement of the cost function is less than a user-defined coefficient. The second phase then starts and aims to subdivide mainly large simplices, located as far as possible from the current best point. This is performed until the improvement of the cost function is less than 1% of the current best solution found. Then the algorithm switches back to phase one. This is repeated until the stopping criterion is reached.

Other enhancements of DIRECT algorithm can be found in the literature such as [Liu et al., 2015, 2017] but due to the structure of the algorithm, no modification so far have been able to solve large scale problems with a dimension greater than 500.

1.6 Parallel metaheuristics

The parallelization of metaheuristics has been popular over the last three decades in the field of optimization. Indeed, several works have mainly focus on adapting existing algorithms to allow them to take profit from multithreaded or multi-nodes environments. In this section, we present the different adaptation of the algorithms presented earlier. The

increase in dimension and complexity of real-world applications are the main motivations for the increasing development of parallel metaheuristics.

1.6.1 Parallel Evolutionary Algorithms

In [Gorges-Schleuter, 1989], the authors focused on parallelizing the well-known metaheuristic called *Genetic Algorithm (GA)*. They developed a parallelized version, called *Parallel Genetic Algorithm (PGA)*. The main idea behind this algorithm is to distribute the selection scheme by making each individual looking for a good solution, but only among its neighbours. This approach obtained satisfactory results on the Travel Salesman problem. More recently, in [Liu & Wang, 2015], a parallel version of the Genetic Algorithm (PGA) is proposed and modified to solve the Generalized Assignment Problem. The main challenge faced by PGA parallel environment is the costly synchronisation at each iteration which increases as more processor are involved. They also observed that the amount of computation required to solve a problem does not dependent only on the problem size. To overcome those challenges, the author proposed an asynchronous migration strategy to enable efficient interactions between sub-populations, and improve the overlapping of computation and communication. Their approach was significantly improved by using a buffer-based communication and non-blocking message. The algorithm is implemented using MPI for communication between nodes.

In, [Gong et al., 2015] a process was proposed to parallelize any EA algorithm. Four components were identified. The first one is the algorithm itself and can represent other population-based algorithms, such as ACO or PSO as they share common features with EAs. The second component refers to the model chosen for the distribution architecture. In addition to the Master-Slave model being the most common one, other models are available such as hierarchical, Cellular/Fined grained or multi-agents. The choice of model has an influences on the programming environment which is the third component. Among them, OpenMP for multi-threads environments, MPI for multi-node clusters or MapReduce and its implementation, *i.e.* Hadoop for the more recent one. Finally, the underlying IT infrastructure is the fourth and final component of the proposed framework. Indeed, this choice as a huge influence weather the distributed algorithm will run on the cloud, on GPUs or full grids environments. In line with this last component, in recent work, a Genetic Algorithm has been specifically designed to run on a cloud computing-based environment using Hadoop [Kečo et al., 2016]. Authors reported an unlimited scalability thanks to the MapReduce framework used and a reduced computation time. In this case, GA was used in conjunction with other modern techniques, *i.e.* Artificial Neural Networks and Support Vector Machines. They applied it to gene selection in cancer classification and justified the use of the Hadoop framework due to

the important size of the data set. A GPU adaptation [Luo et al., 2019] has been proposed as it significantly reduces the computation time. Authors mentioned that their parallel version is highly consistent with the hierarchy of threads and memory of the GPU framework used known as CUDA.

1.6.2 Parallel Ant Colony Algorithm

The *Ant Colony Optimization (ACO)* algorithm has been the subject of many works [Baocheng et al., 2012; Liu & He, 2012; Delisle et al., 2009; Situ et al., 2017]. In their work [Baocheng et al., 2012], the authors run iteratively different sequential ACO [Liu & He, 2012], the parallelization is made at the colony or ant level, searching independently and sending back their results synchronously or asynchronously, depending on the parallel model. The latter has parallelized the algorithm on a multi-core processor environment using OpenMP. They concluded that the execution time can be greatly reduced without losing quality of the final solution. In [Situ et al., 2017], authors studied a parallelized version of the ACO algorithm applied to the Taxi-Passenger Matching. The idea was to divide the city being optimized into several regions to reduce the dimension of the problem. Hence, making the approach similar to a D&C strategy. They explore regions in parallel allowing the algorithm to find a good solution faster.

1.6.3 Parallelized Decomposition methods

As the main approach presented in this thesis is a Divide-and-Conquer based algorithm, the literature on Branch-and-Bound algorithms is also taken into account. In [Herrera et al., 2017], the authors mentioned different strategies to parallelized B&B algorithms: 1. Parallelizing the nodes' evaluation; 2. Parallelizing the construction of the search tree; 3. A combination of the two previous strategies. They studied these three strategies on a multi-thread environment, reaching a linear tendency of the SpeedUp.

Only a few algorithms that use the geometric decomposition of the search space were proposed in the literature. However, when dividing the search domain, both DIRECT and FRACTOP suffer from the exponential growth of subregions, making those algorithms computationally expensive on large-scale problems not applicable for big optimization. In the case of Multiple Optima Sierpinski Searcher, the authors stated that the chosen geometric forms will not allow the algorithm to cover the entire search domain. As DIRECT does not perform well on high dimensions problems, in [He et al., 2004], authors proposed a parallelized version of the algorithm to tackle this issue. To do so, a multi-start strategy was used via evaluating multiple starting points on different processors. The evaluations of the objective function were also distributed among the

different CPUs. This algorithm was implemented using both OpenMP for the multi-threading part and MPI for passing messages over multiple processors. It is known that DIRECT divides the search space into hypercubes, the number of vertices to evaluate grows exponentially, when the dimension of the problem increases making the algorithm computationally expensive on large-scale problems: seventeen (17) hours were necessary to reach *238397* function evaluations using *141* processors. It shows that even parallelized, DIRECT is not suited for large scale problems.

1.7 Multi-Objective Optimization

In this section, we cover a different branch of optimization related to multi-objective optimization. This refers to the type of problems where multiple cost functions are optimized simultaneously and usually contradict each other. Many real-world applications belong are multi-objective problems and some are detailed in [Stewart et al., 2008].

A Multi-Objective Optimization Problem (MOP) can be formulated as follow:

$$\begin{aligned} &minimize\, F(x) = (f_1(x), ..., f_k(x))^T \\ &subject\, to\, x \in X \end{aligned} \tag{1.6}$$

where X is the decision variable space, $F : X \rightarrow \mathbb{R}^k$ is composed of k real-valued objective functions and $\mathbb{R}^k$ is the objective space.

In single-objective optimization, it is possible to compare two given solutions and determine the best one. The results of an algorithm solving this type of problems is a single value aiming to be the global optimum of the given function. However, in multi-objective optimization, there does not exist a straightforward technique to determine the best solution between two given candidates. The method most commonly used to compare solutions in the context of MOP is called "Pareto dominance relation". Instead of a single optimal solution, this leads to a set of different trade-offs among the multiple objectives. The equivalent of the global optimum, in MOP, is called the true Pareto Front (PF). It is the optimum set of values for the different objective functions. Pareto Dominance can be defined as follow. If we consider, z^1, $z^2 \in \mathbb{R}^k$, z^1 is said to Pareto-Dominate z^2, denoted $z^1 \prec_{pareto} z^2$, if and only if both Equations 1.7 and 1.8 are satisfied.

$$\forall i : z_i^1 \geq z_i^2,\ i \in 1, ..., k \tag{1.7}$$

$$\exists j : z_j^1 > z_j^2, \ j \in 1, ..., k \tag{1.8}$$

A point $x^* \in X$ is Pareto Optimal if there is no point $x \in X$ such that $f(x) \prec_{pareto} f(x^*)$.

The set of Pareto Optimal solutions PS^* and its image in the objective space called Pareto Front PF^* are defined respectively in Equation 1.9 and 1.10.

$$PS^* = \{\mathbf{x} \in \mathcal{X} \mid \nexists \mathbf{y} \in \mathcal{X} : \mathbf{f}(\mathbf{y}) \preceq \mathbf{f}(\mathbf{x})\} \tag{1.9}$$

$$\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \mid \mathbf{x} \in \mathcal{P}^*\} \tag{1.10}$$

Many of the well-known metaheuristics have been adapted to solve multi-objective problems such as Genetic Algorithms (GA) or Particle Swarm Optimization (PSO). Two main methodologies exist to develop or adapt algorithms to solve MOPs: Dominance and Scalarization.

1.7.1 Dominance-based algorithms

Dominance-based algorithms use the principle of Pareto-Dominance, defined earlier in Equations 1.7 and 1.8, to sort dominate solutions from non-dominated ones.

1.7.1.1 Particle Swarm Optimization for multi-objective Optimization

In [Sierra & Coello Coello, 2005], the authors proposed a *Multi-Objective Particle Swarm Optimizer* based on Pareto Dominance and the use of the crowding distance. They compared their approach to other PSO-based algorithms for MOP. Both crowding distance and non-domination sorting are used to select the set of new leaders. A more recent adaptation was proposed in [Helbig, 2016]. To increase the diversity of the Pareto Fronts generated, they used an ϵ-dominance method. It allowed solutions with small degradation to be selected as good candidats. In this approach, the global swarm is divided into sub-swarms, each optimizing one objective function. The knowledge is shared among each sub-swarm and the dominance is used to select a particle's new position. From the literature, different studies [Sierra & Coello Coello, 2005; Helbig, 2016], show that PSO-based multi-objective algorithms are sensitive to the adjustment of their attractiveness parameter to avoid early convergence.

1.7.1.2 Non-dominated Sorting Genetic Algorithm (NSGA-II)

The most popular dominance-based algorithm is undoubtedly NSGA-II developed by Deb et al. [2002b]. It is a Genetic Algorithm developed specifically to solve multi-objective problems. It follows the first version NSGA [Srinivas & Deb, 1994] which was one of the first Evolution Algorithm developed to solve MOPs. NSGA-II was developed to overcome the first edition's problems, *i.e.* its high computational complexity of non-dominated sorting; lack of elitism and the need for specifying a sharing parameter. To do so, the second version uses three main mechanisms to find the Pareto Front. It emphasizes the non-dominated solutions to pre-sort individuals in each generation. Then the selection procedure chosen is based on the elitist-preserving principle thereby assuring preservation of previously found good solutions to generate the next generations. Studies have shown that elitist mechanism helps the convergence of Multi-Objective Evolutionary Algorithms (MEOAD) [Zitzler, 1999]. Finally, it ensures diversity among the resulting solutions using a density-estimation metric called the *Crowding-Distance*. This metric estimates the density of solutions surrounding a particular solution in the population. The overall crowding distance is computed as the sum of each individual's distance corresponding to each normalized objective functions.

This algorithm has shown great results compare to existing MOEAs on two-objective functions and shows decreasing results when the number of objective increases. [Deb & Jain, 2014] is an extension of NSGA-II adapted to solve many-objective problems, *i.e.* more than three objectives. It works with a set of supplied or predefined reference points aiming to maintain the diversity among population members.

NSGA-II has attracted the attention of the multi-objective community and has been the subject of many studies and application in the literature. For instance in [Lakshmi et al., 2011], authors have modified the algorithm and applied it to the economic and emission dispatch problem. In [Deb et al., 2007], NSGA-II is applied to the hydro-thermal power scheduling problem. Other work showed NSGA-II applied to a reactive power compensation problem in [Pires et al., 2012].

1.7.2 Scalarization in Multi-objective optimization

The main idea behind scalarization is to transform a multi-objective problem into a mono-objective one by aggregating the different objective functions. Here, each objective function has a weight coefficient and the objective is to minimize the sum of all weighted objective functions (in case of minimisation). Several scalarization methods can be found in the literature [López Jaimes & Zapotecas-Martínez, 2011].

1.7.2.1 Scalarization techniques

Weighted Sum

This approach consists of using a weight vector $\omega = (\omega_1, ..., \omega_k)$, to combine the k objective functions as follow:

$$\begin{aligned} &minimize \sum_{i=1}^{k} \omega_i f_i(x) \\ &subject\ to\ x \in X \end{aligned} \tag{1.11}$$

with $\omega_i \geq 0\ for\ i = 1, ..., k$ and $\sum_{i=1}^{k} \omega_i = 1$. The set of non-dominated solutions can be generated by using different weight vectors ω in using the weighted sum approach. In the case where the Pareto Front is convex (or concave in case of maximization), this technique works well [Zhang & Li, 2007]. However, it is not always the case when optimizing multi-objective problems.

Tcheybycheff method

This technique [Miettinen et al., 2008] has the particularity to introduce the notion of ideal point or reference point z_i^* is as follow:

$$\begin{aligned} Minimize \quad & \max_{i=1,...,k} \quad [\omega_i(f_i(x) - z_i^*)] \\ \text{Subject to} \quad & x \in X \end{aligned} \tag{1.12}$$

with k the number of objective functions to optimise, $z^* = (z_1^*, ..., z_k^*)$ the reference point with z_i^* the optimum of function f_i and as in the previous method, the weight vector $\omega = (\omega_1, ..., \omega_k)$ with $\omega_i \geq 0\ for\ i = 1, ..., k$ and $\sum_{i=1}^{k} \omega_i = 1$. The major problem with this technique is that the aggregation obtained with the vector ω is not smooth for a continuous problem [Zhang & Li, 2007].

Augmented weighted Tchebycheff

In [Steuer & Choo, 1983], the authors proposed a modified version of the Tchebycheff as shown on the Equation 1.13:

$$\begin{array}{ll} \text{Minimize} & \max_{i=1,\ldots,k} \{w_i \, |f_i(\mathbf{x}) - z_i^\star|\} + \rho \sum_{i=1}^{k} |f_i(\mathbf{x}) - z_i^\star| \\ \text{subject to} & \mathbf{x} \in \mathcal{X} \end{array} \tag{1.13}$$

with ρ being a small positive scalar. However, this technique first integrates another parameter and, the authors in [López Jaimes & Zapotecas-Martínez, 2011] indicate that using that approach some Pareto optimal solutions cannot be found.

ϵ-Constraint Method

In this approach the different objectives are not aggregated, however when one objective is minimized, the other is used as constrained bound by some acceptance level ϵ.

$$\begin{array}{lll} \text{Minimize} & f_l(\mathbf{x}) & \\ \text{subject to} & f_i(\mathbf{x}) \leq \varepsilon_i \quad \forall i = 1, \ldots, k \quad i \neq l \\ & \mathbf{x} \in \mathcal{X} \end{array} \tag{1.14}$$

To find the Pareto Front, the problem as formulated in 1.14 needs to be solved using multiple different values for ϵ_i. The range of the reference objective and increment for the constraints imposed by ϵ need to be both provided by the user. This increment determines the number of Pareto optimal solutions generated. To build the final Pareto Front, k single objective problems need to be solved.

1.7.2.2 Scalarization-Based Algorithms

Multi-objective Evolutionary Algorithm referred in the literature as MOEA has been popular in solving MOPs. Where NSGA-II is a dominance-based approach, other EA algorithms have used scalarization techniques to solve MOP and are called decomposition methods. They differ from algorithms presented in Section 1.5 because they decompose the objective space and not the search space.

One of the well-known frameworks for EA using decomposition is called MOEOA/D and is proposed in [Zhang & Li, 2007]. It uses scalarization to decompose the MOP into multiple scalar optimization subproblems and solve optimizes them simultaneously by evolving a population of candidate solutions. Subproblems are solved using information from the neighbouring subproblems making this approach less computationally expensive than MOGLS [Jaszkiewicz, 2002] (an older EA to solve the 0/1 knapsack problem) and NSGA-II. The authors argued that domination does not define a complete ordering

among the candidate solutions in the objective space. As the purpose is to find a well-diversified Pareto Front, conventional selection operators designed originally for scalar optimization cannot be used in the dominance-based algorithm. Using scalarization, individuals can be assigned a fitness value to help the selection process and therefore, operators designed for single-objective problems, which have been extensively tested in the literature can be used. Different scalarization techniques can be used but the authors in [Zhang & Li, 2007] chose the Tcheybycheff method functions as it shows better results than other methods. In this technique, the diversity of solutions can be done using properly distributed weights vectors in the scalarization methods. Where generating weights vector in 2-objective functions is straightforward, techniques have been developed in the literature to deal with functions having more than 3 objectives. MOEA/D uses uniformly distributed weight vectors and use the Euclidean distance to measure the closeness between two vectors.

This approach shows competitive results compared to NSGA-II on both discrete and continuous problems. However, MOEA/D is sensitive to its control parameter T being the number of weight vectors in the neighbourhood considered around of each weight vector and results show that the approach does not work well when T is too small. The size of the population is also an important parameter in any EA that impacts significantly the performance of MOEA/D.

This work has inspired many other algorithms in the literature to use scalarization techniques combines with Evolutionary Algorithms to solve multi-objective problems. For instance, [Saborido Infantes et al., 2017] presents GWASFGA which stands for "Global Weighting Achievement Scalarizing Function Genetic Algorithm". This algorithm is also based on the scalarization method and uses achievement scalarizing function which based on the Tcheybycheff method but includes the use of the utopian and the nadir points. GWASFGA generates the weight vectors so that they define an evenly distributed set of projection in the objective space. A more recent work, CDG [Cai et al., 2018] is also a decomposition-based MOEA. Instead of using a traditional scalarization method such as Tcheybycheff, CDG-MOEA uses a constrained decomposition with grids. One objective function is selected to be optimized while the other objective functions are converted into constraints by setting up the upper and lower bounds.

One can remark that Evolutionary Algorithms have been the main focus in MOP using scalarization methods. However, some other works have attempted to adapt traditional metaheuristics such as PSO or Simulated Annealing. SA have been adapted to use the weighted sum in [Loukil et al., 2007] for solving the production scheduling problem. In [Vazan & Cervenanska, 2018], authors applied different scalarization techniques to SA but results showed that the approach could only solve partially the studied problems.

Finally, Particle Swarm Optimization combined with scalarization has also been studied. A Weighted approach was used in [Lee et al., 2014]. This solution showed interesting results compared to NSGA-II but only the hypervolume metric was used to compare the approaches. Comparing to algorithms in the context of MOP involves many metrics and multiple ones should be used to properly conduct a comparison analysis.

1.7.3 Performance evaluation in MOP

In mono-objective problems, comparing algorithms is straightforward. The lower the solution outcome is (in minimization), the better the algorithm is (provided that the stopping criterion is set up to be the same among compared algorithms). In the context of multi-objective, this task is not trivial as the outcome is composed of multiple Pareto Optimal solutions, therefore one has to find a metric allowing to compare Pareto Fronts. In the literature different metrics have been developed to measure the quality of the solution sets obtained by different algorithms. In [Riquelme-Granada et al., 2015], the authors have referenced 54 different metrics found in the literature. Each metric measures different characteristics of a Pareto Front. Those characteristics can be classified into three main categories [Okabe et al., 2003]:

- *Convergence* (or accuracy), *i.e.* the closeness from the theoretical Pareto Front;
- *cardinality*, *i.e.* the number of points in the front;
- *diversity*, *i.e.* the distribution of the front. The points in a Pareto Front should be well spread and not concentrated around one area of the objective space.

As mentioned, to measure those different aspects, multiple metrics exists in the literature. We have chosen to focus on the four most commonly used as combined, they allow measuring all aspects of a Pareto Front:

- *The Hypervolume*, being the most used and the only one measuring the three focused aspects, it is a must have. As a recall, it measures the size of the portion of the objective space that is dominated by an approximation set (Figure 1.5).
- *The Generational Distance* metric (GD), computes the average distance from a set of solutions obtained by an algorithm to the true Pareto-Front.
- *The Inverted Generational* Distance (IGD), measures both convergence and diversity by computing the distance from each point known in the true Pareto-Front to each point of a set of solutions found by an algorithm.

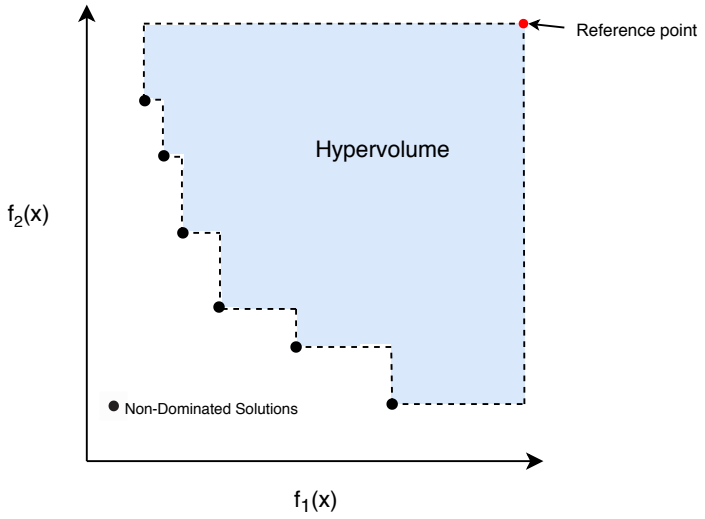


FIGURE 1.5: An illustration of the hypervolume with regards to the nadir point.

- *The Spread*, this metric measures the extent of the spread achieved among the obtained solutions. It measures how well spread the non-dominated solutions are over the objective space.

Aiming to find a good Pareto Front, it is important to mention that both while the Hypervolume aims to be maximized, the other metrics aim to be minimized.

1.7.3.1 The Hypervolume

The hypervolume was originally used in [Zitzler & Thiele, 1999] and [Zitzler & Thiele, 1998] to quantify the Pareto Front generated by different MOEAs. Originally it was referred to as "Size fo the space covered". The original denotation illustrates perfectly what it measures, *i.e.* the space covered by the Pareto Front in the objective space. Figure 1.5 illustrates the hypervolume in the case of a two-objective problem. In the case of minimization, the aim is to maximize the hypervolume (blue area). This is because the higher the hypervolume, the further solutions are from the z^{nad}, hence the better they are. Hypervolume is known to be the most used performance metrics in MOP [Riquelme-Granada et al., 2015] and the only one measuring accuracy (closeness to the true PF), diversity (the spread of solutions) and cardinality (number of solutions). The hypervolume does not require any prior knowledge to be computed. However, this metric is computationally expensive. Its worst case complexity of the hypervolume is exponential with regards to the number of objective functions.

1.7.3.2 The Generational Distance and Inverted Generational Distance

The Generational Distance (GD) is the second most used metrics [Riquelme-Granada et al., 2015] and measures the accuracy of an approximated Pareto Set A with regards to the true Pareto Front PF^*. In other words, it measures *"how far"* A is from PF^*. GD measures the average (using Euclidean distance) between the solutions of A and the nearest member of PF^*. The GD is defined as in Equation 1.15:

$$GD\left(A, P^*\right) = \frac{\sqrt{\sum_{v \in A} d(v, PF^*)}}{|S|} \tag{1.15}$$

with $d(v, PF^*)$, the Euclidean distance between $v \in A$ and the closest member in PF^*.

Where the GD measures the distance of A from PF^*, its inverted variation, the Inverted Generational Distance, measures the distance from PF^* to A. It uses the minimum Euclidean distance instead of the average one. The IGD is defined in Equation 1.16.

$$IGD\left(P^*, A\right) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \tag{1.16}$$

with $d(v, A)$, the minimum Euclidean distance between a member in PF^* and the closest member $v \in A$. GD only measures the accuracy of A, however, IGD measures both diversity and convergence. If $IGD\left(P^*, A\right)$ is low, this means that A must be both close to PF^* and cover it enough not to miss any part of the true Pareto Front. Both GD and IDG metrics have a low computational cost (particularly compared to the hypervolume). However, they require the true Pareto Front.

A more recent version of this metric called IGD+, has been proposed in [Ishibuchi et al., 2015]. The main advantage of this metric is that it is weakly Pareto compliant. The Pareto dominance is taken into account when the Euclidean distance between a solution point and a reference point is calculated.

1.7.3.3 The Spread

The Spread, also referred as the $\Delta - metric$ measures the diversity of the Pareto Set generated by an algorithm. It has been shown in [Deb et al., 2002b] and is defined in Equation 1.17:

$$\Delta(A) = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left|d_i - \overline{d}\right|}{d_f + d_l + (N-1)\overline{d}} \tag{1.17}$$

with A being the Pareto Front found by the algorithm, d_f and d_l being the Euclidean distance between the extreme solutions of the true Pareto Front PF^* and the boundaries solutions in A. $\overline{d}$ is the average distance between $d_i, i = 1, 2...., (N-1)$, provided they are N solutions in A. As an indication, a good spread of solutions would make all distances $d_i = \overline{d}$ and $d_f = d_l = 0$.

As this metric originally only works for 2-Objective problems, Aimin Zhou et al. [2006] have developed a Generalized version for 3-Objectives and more as in Equation 1.18.

$$\begin{gathered}\Delta\left(A, PF^*\right) = \frac{\sum_{i=1}^{m} d(e_i,A)+\sum_{X\in A^*}|d(X,A)-\overline{d}|}{\sum_{i=1}^{m} d(e_i,A)+|PF^*|\overline{d}} \\ \text{where } \{e_1,\ldots,e_m\} \text{ are } m \text{ extreme solutions in } PF* \text{ and} \\ d(X,A) = \min_{Y\in A, Y\neq X}\|F(X)-F(Y)\|^2 \\ \overline{d} = \frac{1}{|PF^*|}\sum_{X\in PF^*} d(X,A)\end{gathered} \tag{1.18}$$

In the rest of this document both the Spread and its Generalized version will be referred "Spread". This metric has a low computational cost and does not required any prio knowledge to be computed.

1.8 Conclusion

In this chapter, we have presented a brief overview of the field of optimization followed by a focus on MSO optimization and metaheuristics. The main algorithms in each family were detailed such as Simulated Annealing for Single-Solution Optimization, Evolutionary Algorithm as population-based algorithms and finally the metaheuristics such as DIRECT which decomposes the search space aiming to find the global optimum.

In our work, we are interested in the design of a decomposition-based metaheuristic with a geometrical form that would both fully cover the search space and is suitable for scaling up, allowing to solve large scale optimization problems. This new approach will be presented in the next chapter.

With the increase of problems' complexity and the potential provided with modern IT architectures, we proposed a new approach to benefit from both multi-thread and multi-node environments aiming to improve speed and accuracy as well as keeping the implementation as simple as possible. This work will be presented on Chapter 3.

The proposed decomposition approach was adapted to MOP using both dominance and scalarization in Chapter 4. To evaluate performances we have decided to select multiple metrics as the literature shows that each one measure different characteristics of the resulting Pareto Front.

Chapter 2

Design of Fractal Decomposition based Algorithm

2.1 Introduction

In this chapter, we present the new metaheuristic called "Fractal Decomposition based Algorithm" (FDA). FDA [Nakib et al., 2017] is based on a fractal geometrical decomposition of the search space. Indeed, the main principle of the approach consists of dividing the feasible search space into subregions with the same geometrical pattern. Hyperspheres were chosen as geometrical form as it has the benefit of scaling well as the dimension of the problem increase. As pointed out before, the current methods in the literature do not scale well when the problems' dimension increases. At each iteration, the most promising subregions are selected and further decomposed by FDA. This approach tends to provide a dense set of samples and has interesting theoretical convergence properties. This work aims to propose a new algorithm based on this approach with low complexity and which performs well in case of large-scale problems. To do so, a low complex method, that profits from fractals properties is proposed.

In addition, the following motivations were taken into account when developing FDA:

- FDA has been designed to be easily implemented.
- FDA does not need any analytical information on the functions being optimized.
- FDA is a deterministic algorithm.

A performance analysis is conducted on large-scale global optimization test functions with dimensions going from 50 to 1000. The obtained results, and the comparison with

other metaheuristics designed to solve the same types of problems, show the efficiency and the competitiveness of the proposed algorithm to solve large scale optimization problems. It should be noted that the proposed algorithm is a single solution-based metaheuristic while other competing algorithms are population-based metaheuristic.

The rest of this chapter is organized as follows. Section 2.2 presents the principle of the geometric fractal decomposition, while, the coverage of the search space is discussed in Section 2.3. The proposed algorithm is detailed in Section 2.4, obtained results are pointed out and discussed in Section 2.5.

2.2 Geometric Fractal Decomposition

In this work, a geometrical fractal decomposition based on hypersphere as an elementary geometric form is considered. This choice was motivated by its low complexity and flexibility to cover a part of the search space. Indeed, it is easy to go from one center of a hypersphere to another analytically without storing them.

An example of a geometric fractal decomposition is presented in Figure 2.1, where an example of a fractal dimension of four is illustrated for four levels of decomposition.

Indeed, there are many schemes to divide an N-dimension search space. Several methods were tested as the Descartes theorem but the complexity at the generalization to the N-dimensional increases exponentially. As the goal is to find a scalable method that allows the whole search space to be decomposed, we propose to use overlapped hyperspheres. Then, a geometric decomposition without central hyperspheres was considered. Indeed, such a recursive division of the search space with a fixed number of hyperspheres at each level is called a fractal decomposition and, the number of hyperspheres inside a hypersphere can be seen as *the fractal dimension*.

The advantages of this decomposition are summarized in the following propositions:

Proposition 1. When $2 \times D$ equal hyperspheres are inscribed within a bigger hypersphere in an D-dimensional space, then the ratio (δ) between their radii does not depend on D and is equal to $\delta = (1 + \sqrt{2})$.

The proof of this proposition can be easily done using geometric properties of hyperspheres.

Proposition 2. The hyperspheres fractal decomposition allows the centers and radius of the hyperspheres to be found analytically.

Proof. To decompose the search space, the biggest hypersphere within the search domain is used with a center $\vec{C}^{(1)}$ and its radius r obtained using expressions (2.1) and (2.2), respectively.

$$\vec{C}_j^{(1)} = L + (U - L)/2, \text{ for } j = 1, 2, \ldots, N \tag{2.1}$$

where $\vec{C}^{(1)}$ is the coordinate of the center of the biggest hypersphere within the search space, and r is its radius.

$$r = (U - L)/2 \tag{2.2}$$

where U is the upper bound, L is the lower bound of the whole search space, and D is the dimension of the space.

So, in case of dividing the search space into $2 \times D$ equal hyperspheres, the position of centers $\vec{C}_j^{(i)}$, and their radii r' are given by :

$$\vec{C}_j^{(i)} = \vec{C}_j^{(i)} + (-1)^i \times \left((r - r') \times \vec{e}_j\right) \tag{2.3}$$

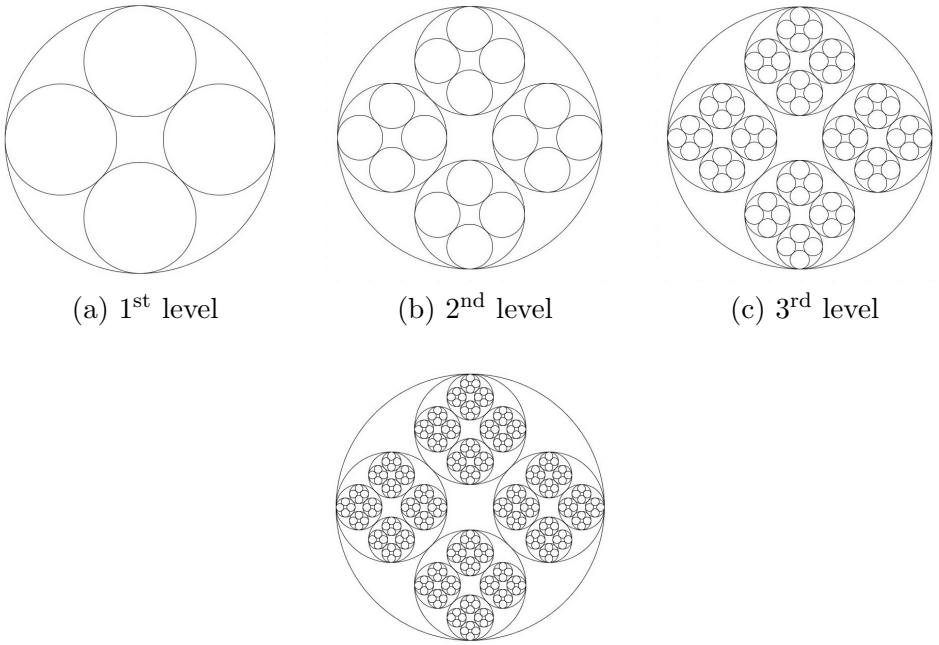
where $\vec{e}_j$ is the unit vector at the dimension j, and i is the index of the hypersphere $i = 1, \ldots 2\ D$, and $r' = r/\delta$.

□

2.3 Coverage of the search space via the Fractal Decomposition

As it can be seen on Figure 2.1, the geometric fractal decomposition does not cover all the space. Thus, to overcome this problem, an increase of the radius (inflation) of each hypersphere of the following level must be performed. This increase produces overlaps between hyperspheres and, thus, allows all the space to be covered as shown on Figure 2.2. So, the ratio between the inflated and the original radii of the hyperspheres, called relaxation coefficient, is named α.

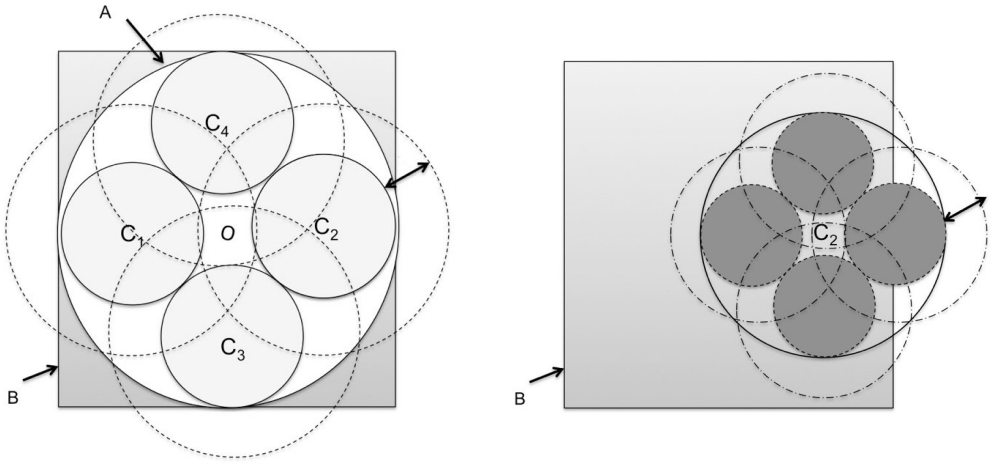
In the following, the expression giving the value of the lower bound of this coefficient α is presented.



(a) 1st level (b) 2nd level (c) 3rd level

(d) 4th level

FIGURE 2.1: Illustration of the fractal decomposition of the search space : the depth of the decomposition equal to 4.



(a) Geometric decomposition at level 1 (b) Geometric decomposition at level 2

FIGURE 2.2: Illustration of the decomposition procedure in the case of a 2D search space, where A is the biggest hypersphere inside the search space (B), C_1, C_2, C_3, and C_4 are centers of hyperspheres at the first level.

2.3.1 Relaxation at the first level

In this section the value of the relaxation coefficient (α) at the first level is calculated.

Denote the radius of parent hypersphere r, and the radius of child hyperspheres r'.

Let us consider the centers of small hyperspheres are located in points:

$$O_i^{\pm} = (0, \ldots, 0, \pm d, 0, \ldots, 0) \tag{2.4}$$

where the only non-zero entry $\pm d$ is in i-th position, and $d = r - r' = r(1 - \frac{1}{1+\sqrt{2}})$.

The minimal required relaxation coefficient value α_n s.t. any arbitrary point from inside of parent-hypersphere is covered by one of the inflated (relaxed) child-hyperspheres with radius $r'' = \alpha_n r'$. To do this, we first estimate the inflated radius r_A for an arbitrary fixed point A, and then r'' will be defined as $r'' = \max\limits_A r_A$. Let a point $A(x_1, \ldots, x_D)$ lie inside the parent-hypersphere, *i.e.* $x_1^2 + x_2^2 + \ldots + x_D^2 \leq r^2$. The minimal required value of the radius r_A to cover A is defined by:

$$\begin{aligned} r_A^2 &= \min_{i,\pm} \|O_i^{\pm} A\|^2 \\ &= \min_{i,\pm} \left(\sum_{j \neq i} x_j^2 + (|x_i| \pm d)^2 \right). \end{aligned} \tag{2.5}$$

As $|x_i| \geq 0$, and $d > 0$, then $(|x_i| - d)^2 < (|x_i| + d)^2$. Hence, $r_A^2 = \min\limits_i \left(\sum\limits_{i \neq j} x_j^2 + (|x_i| - d)^2 \right)$ (we denote $a^2 = \|OA\|^2 = \sum\limits_{i=1}^{n} x_i^2$), then:

$$\begin{aligned} r_A^2 &= \min_i \left(\sum_{i \neq j} x_j^2 + (|x_i| - d)^2 \right) \\ &= a^2 - d(2 \max_i |x_i| - d). \end{aligned} \tag{2.6}$$

As A was considered as an arbitrary fixed point, to cover all points inside the hypersphere, we need to maximize the relaxation of the radius r_A over A. Thus:

$$\begin{aligned} r''^2 &= \max_{\|OA\|^2 \leq r^2} r_A^2 \\ &= \max_{0 \leq a \leq r} \max_{\|OA\| = a} (a^2 - d(2 \max_i |x_i| - d)) \end{aligned} \tag{2.7}$$

Now, we show that the point A cannot be a *maximum point* unless all of its coordinates are equal up to the sign. Without loss of generality, we assume that $x_i \geq 0$ for all $1 \leq i \leq D$ (indeed, the sign of x_i does not influence the value of the considered function under *max* operator). Moreover, assume that there exists an index s such that $x_s \neq \max\limits_k x_k$. Let $i_1, i_2, \ldots, i_m$ be the indices of the highest coordinates of A, and j be the entry of

the second highest coordinates:

$$x_{i_1} = x_{i_2} = \ldots = x_{i_m} = \max_k x_k \tag{2.8}$$
$$x_j = \max_{k \neq i_1, i_2, \ldots, i_m} x_k$$

Under this assumption it is possible to *shift* the point A along the sphere $\|OA\| = a$ such as the maximized function is increasing, and thus the point A cannot be a point of maximum.

We Consider another point, A', with coordinates $(x'_1, \ldots, x'_D)$, where:

$$x'_k = x_k, \quad k \neq i_1, i_2, \ldots, i_m, j, \tag{2.9}$$
$$x'_i = x_i - \delta^-, \quad i = i_1, i_2, \ldots, i_m, \tag{2.10}$$
$$x'_j = x_j + \delta^+, \tag{2.11}$$

and $0 < \delta^-, \delta^+ < \frac{1}{2}(x_{i_1} - x_j)$ are such that $\|OA'\|^2 = \|OA\|^2 = a^2$. Then:

$$x'_i = x_i - \delta^- > x_j + \delta^+ = x'_j, \quad i = i_1, i_2, \ldots, \tag{2.12}$$
$$x'_j = x_j + \delta^+ > x_j \geq x_k = x'_k, \quad k \neq i_1, i_2, \ldots, i_m, j. \tag{2.13}$$

Hence, $x'_{i_1} = x'_{i_2} = \ldots x'_{i_m} = \max_k x'_k$. Denote $f(A) = (a^2 - d(2 \max_k |x_k| - d))$, where as before $a^2 = \|OA\|^2$. Then:

$$\begin{aligned} f(A') &= (a^2 - d(2x'_{i_1} - d)) \\ &= (a^2 - d(2x_{i_1} - 2\delta^- - d)) \\ &= (a^2 - d(2x_{i_1} - d)) + 2\delta^- d \\ &> f(A). \end{aligned} \tag{2.14}$$

In other terms:

$$\arg\max_{\|OA\|=a} (a^2 - d(2 \max_i |x_i| - d)) \neq A. \tag{2.15}$$

One can also remark that the maximum exists. Indeed, function $f(A)$ is continuous, and can be considered as a function on a compact of lower dimension, defined by equality $\|OA\| = a$. Hence, by extreme value theorem, it reaches its maximum, and based on the inequality above, we infer that the point of maximum has to have equal coordinates (up to the sign). In particular, the maximum is reached at the point A^* having coordinates

$(\frac{a}{\sqrt{D}}, \ldots, \frac{a}{\sqrt{D}})$. Back to the expression for the relaxation radius r'', we get:

$$\begin{aligned} r''^2 &= \max_{0\le a\le r} \max_{\|OA\|=a} (a^2 - d(2\max_i |x_i| - d)) \\ &= \max_{0\le a\le r} \left(a^2 - \frac{2ad}{\sqrt{D}} + d^2\right). \end{aligned} \tag{2.16}$$

The quadratic function of a under the max operator reaches its maximum on one of interval ends. Denote $g(a) = a^2 - \frac{2ad}{\sqrt{D}} + d^2$. Then:

$$g(0) = d^2 = \lambda^2 r^2, \tag{2.17}$$

$$g(r) = r^2 - \frac{2rd}{\sqrt{D}} + d^2 = r^2 - \frac{2\lambda r^2}{\sqrt{D}} + \lambda^2 r^2, \tag{2.18}$$

where $\lambda = 1 - \frac{1}{1+\sqrt{2}} = \frac{\sqrt{2}}{1+\sqrt{2}} \approx 0.59$. From this, it is obvious to see that for $D \ge 2$

$$\frac{2\lambda}{\sqrt{D}} < 1$$

, and so $g(r) > g(0)$. Finally:

$$\begin{aligned} r''^2 &= \max_{0\le a\le r} \left(a^2 - \frac{2ad}{\sqrt{D}} + d^2\right) \\ &= g(r) \\ &= r^2 - \frac{2\lambda r^2}{\sqrt{D}} + \lambda^2 r^2 \\ &= r^2 \left(1 - \frac{2\lambda}{\sqrt{D}} + \lambda^2\right), \end{aligned} \tag{2.19}$$

and, thus:

$$= \sqrt{5 + 2\sqrt{2} - \frac{2\sqrt{2}(1+\sqrt{2})}{\sqrt{D}}}. \tag{2.20}$$

If one would like to set the unified relaxation coefficient, it would have to be:

$$\alpha^{(1)} = \sup_{D\ge 1} \alpha_D = \sup_{D\ge 1} \sqrt{5 + 2\sqrt{2} - \frac{2\sqrt{2}(1+\sqrt{2})}{\sqrt{D}}} = \sqrt{5 + 2\sqrt{2}} \approx 2.80. \tag{2.21}$$

2.3.2 Lower bound estimation of α

In the previous paragraph, we pointed out the 1^{st}-level. During the proof we established that the most distant point of initial hypersphere to the centers of 1^{st}-level hyperspheres is the point M $(\frac{r}{\sqrt{n}}, \frac{r}{\sqrt{D}}, \ldots, \frac{r}{\sqrt{D}})$, given here up to the sign of indices (because the

symmetry does not influence the distance from the given point to the set of hyperspheres centers). We will now assume that this point is the most distant point for higher levels of decomposition as well. Even if it is not the case, we will obtain the lower bound for the relaxation coefficient, as point the M still needs to be covered.

Let us find, among a k-level decomposition hyperspheres centers', the closest one to the point M. Consider the process of recursive fractal division where at each level, we consider only one of the obtained hyperspheres as a process of approaching the point M. This procedure has a finite number of steps, which are allowed to be made only along one of the axes. The most effective strategy would be then to move along the axis by which the current position and the target point have the biggest difference. Indeed, without loss of generality, we assume that our goal is to approach the point $Q\ (x_1, x_2, \ldots, x_n)$ from point $O\ (0, 0, \ldots, 0)$ with a step of length s. Assume that after this step, we moved to the point $O_i^{\pm}$, where $OO_i^{\pm} = \pm s e_i$. Then:

$$\begin{aligned} d^2(O_i^{\pm}, Q) - d^2(O, Q) &= (x_i - (\pm s))^2 - x_i^2 \\ &= s^2 - 2(\pm s)x_i. \end{aligned} \tag{2.22}$$

In the previous expression (2.22) the first element does not depend on i, while, the second one is the smallest and yields the smallest value (*i.e.* biggest absolute value) when $|x_i|$ is the largest possible and $\operatorname{sign} \pm s = \operatorname{sign} \pm x_i$ (*i.e.* the step was made in the direction of x_i, not the opposite one).

Denote the radius of the parent-hypersphere $r = r_0$ and the radius of child-hyperspheres of l-level decomposition as r_l. The step on level l is then equal to $s_l = r_{l-1} - r_l$. Then,

$$r_l = \beta r_{l-1}, \tag{2.23}$$

where

$$\beta = \frac{1}{1 + \sqrt{2}}. \tag{2.24}$$

Thus:

$$s_l = r_{l-1} - r_l = \beta^{l-1} r - \beta^l r. \tag{2.25}$$

and

Consider the case $n \geq l$ according the approaching strategy given above, one of the points close, (O_l) to the point M will have the following coordinates:

$$(s_1, s_2, \ldots, s_l, 0, \ldots, 0) = (r - \beta r, \beta r - \beta^2 r, \ldots, \beta^{l-1} r - \beta^l r, 0, \ldots, 0). \tag{2.26}$$

Then, the squared distance between the points O_l and M, *i.e.* the distance we need to cover with the inflated hypersphere, is equal to:

$$\begin{aligned} d^2(O_k, M) = (r - \beta r - \frac{r}{\sqrt{n}})^2 + (\beta r - \beta^2 r - \frac{r}{\sqrt{n}})^2 + \ldots \\ + (\beta^{l-1} r - \beta^l r - \frac{r}{\sqrt{n}})^2 + (n - l)\frac{r^2}{n} \end{aligned} \tag{2.27}$$

Then, the relaxation coefficient $\alpha_n^{(l)}$ is thus equal to:

$$\begin{aligned} \alpha_n^{(l)} &= \frac{d(O_l, M)}{r_l} = \frac{d(O_l, M)}{\beta^l r} \\ &= \sqrt{\sqrt{2}\left((1+\sqrt{2})^{2l} - 1\right) + 1 - \frac{2(1+\sqrt{2})^l\left((1+\sqrt{2})^l - 1\right)}{\sqrt{n}}} \end{aligned} \tag{2.28}$$

Now consider the case $n < l$. Then, one of the closest points to the point M is the point O_l with coordinates:

$$(s_1, s_2, \ldots, s_n + \ldots s_l) = (r - \beta r, \beta r - \beta^2 r, \ldots, \beta^{n-1} r - \beta^l r). \tag{2.29}$$

Similarly to the previous case:

$$d^2(O_l, M) = (r - \beta r - \frac{r}{\sqrt{n}})^2 + (\beta r - \beta^2 r - \frac{r}{\sqrt{n}})^2 + \ldots + (\beta^{n-1} r - \beta^l r - \frac{r}{\sqrt{n}})^2 \tag{2.30}$$

Then, the relaxation coefficient $\alpha_n^{(l)}$ is equal to:

$$\begin{aligned} \alpha_n^{(l)} &= \frac{d(O_l, M)}{r_l} \\ &= \sqrt{\sqrt{2}((\delta^{2l} + c^{2l-2n+1}) + 1 - 2\delta^{l-n+1} - \frac{2}{\sqrt{n}}\delta^l(\delta^l - 1)} \end{aligned} \tag{2.31}$$

where $\delta = (1 + \sqrt{2})$ defined in property 1.

Consequently, combining both cases, we obtain the following formula:

$$\alpha_n^{(l)} \geqslant \begin{cases} eq(2.28),\ n \geq l, \\ \\ eq(2.31),\ \text{otherwise} \end{cases} \tag{2.32}$$

2.4 Proposed Fractal Decomposition based Algorithm

In this section, the proposed algorithm called FDA, that profits from the fractal decomposition, is presented. To find the global optimal solution (if it is known), the obvious way is to explore exhaustively all inflated last level child-hyperspheres, however, it is too time-consuming. To overcome this problem, two heuristics were proposed: the first one, called *promising hypersphere selection heuristic*: it allows selecting the most promising hypersphere for further decomposition. This heuristic is used during the exploration phase. The second heuristic is performed at the last level, called intensification local search heuristic (ILS): its aim is finding the best solution inside a reduced subregion. This second heuristic is used for the intensification phase.

Moreover, as of the proposed fractal decomposition, there is no need to save all information about visited hyperspheres by FDA: all decomposition can be reconstructed analytically. Then, only the best positions met are saved. Indeed, the expression (2.3) is used to compute centers' positions without any past position.

An overview of the proposed algorithm is presented in Algorithm 5. As pointed out before, it uses the hypersphere H as a geometrical form to represent the search space which is repetitively divided into $2 \times dimension$ child-hyperspheres CHs_i where $i = 1, \ldots, 2D$. This decomposition choice is explained in the Proposition 1. Then, the quality q_i^l of each child-hypersphere CHs_i is evaluated using the procedure described in Section 2.4.1. Afterwards, the child-hyperspheres are sorted and, that with the best quality is chosen to be the next hypersphere to be visited (decomposed). This procedure allows the algorithm to guide the search to the most promising region and lead the optimization to start at the best position.

The subregion of the search space limited by a child-hypersphere CHs_i is defined by:

$$CHs_i = \{x \in \Re^D : x - r^{(i)} \leq x \leq x + r^{(i)}\} \tag{2.33}$$

with $i \in L^m$, where $r^{(i)}$ is the radius of the child-hypersphere i, and L^m is the set of indices that constitute the hypersphere at level m, respectively. At each iteration, the most promising child-hypersphere is selected for further decomposition.

Once the last level, called the fractal depth (k), is reached, the intensive local search procedure (ILS) is applied to the sorted CHs_i.

When all of the hyperspheres of the last level are visited, the search is raised up, using a moving-up procedure (Algorithm 7), to another region via the previous depth (level), by replacing the current hypersphere (H) with the following child-hypersphere CHs_i from the sorted list. The process is repeated until one of the stopping criterion is reached or the child-hyperspheres from all the levels were visited.

As it can be noticed, the proposed approach can be compared to the depth-first branch and bound technique (B&B) often used in combinatorial optimization. The main difference is that in our case each branch represents a part of the search space rather than a singular part of the whole solution. Compared to the B&B, some areas are split and, areas that do not seem to be hopeless are further investigated while the most promising one is searched more intensively.

2.4.1 Promising hypersphere selection (Exploration strategy)

This procedure aims to select the most promising region that might contain the best solution or the global optimum. To do so, each hypersphere i created by the decomposition procedure is evaluated.

It is important to mention that each time a solution is evaluated during the exploration phase, a track of the best solution (*BestSol*) and its coordinates (*bestPosition*) is saved.

The first step, to evaluate the hypersphere quality, is to generate two points $\vec{s}_1$ and $\vec{s}_2$ following the expression (2.36) and (2.37). Then, for positions $\vec{s}_1$, $\vec{s}_2$ and the center of the current hypersphere $\vec{C}^l$, their fitnesses f_1, f_2 and f_c, respectively, are calculated as well as their corresponding distances to the best position found so far (BSF) via the Euclidean distance. The last step consists of computing the slope at the three positions ($\vec{s}_1$, $\vec{s}_2$ and $\vec{C}^l$), referred as g_1, g_2 and g_c. This is performed by taking the ratio between the fitness (f_1, f_2 and f_c) and their corresponding distances. Then, the quality for the current hypersphere will be represented by the highest ratio among g_1, g_2 and g_c, denoted by q:

$$q = \max\{g_1, g_2, g_c\} \tag{2.34}$$

Algorithm 5: FDA Algorithm

Input: Deep of the fractal decomposition: $k = 5$ and the tolerance threshold: $\omega_{\min} = 10^{-20}$

Input: Coefficient step-size: $\lambda = 0.5$, inflation coefficient: $\alpha = 1.75$ and dimension of the problem: D

Initialization of the current hypersphere H and the number of function evaluations $NBEval = 0$

Initialize the center $\vec{C}$ at the center of the search space.

Evaluate the objective function of the center $\vec{C}$, initialize the best solution $BestSol$ with the resulting value and the best position $bestPosition$ with $\vec{C}$;

NBEval = NBEval + 1

Calculate the radius r using eq. 2.2; Initialize the level variable: $l = 1$

while *Stopping criteria are not reached* **do**

  Decompose the current hypersphere H using the Fractal procedure using expression (2.3)

  for $2 \times D$ *l-level hypersphere* **do**

    Apply the promissing hypersphere selection procedure described in Section 2.4.1

  end

  Sort the $2 \times D$ hyperspheres at the current l-level

  Replace the current hypersphere H by the first of the sorted hyperspheres at the current level

  if $l == k$ **then**

    for *Each* $2 \times D$ *of* k^{th}*-level hypersphers* **do**

      Apply the ILS heuristics described in Section 2.4.3

    end

    if *stopping criterion is not reached* **then**

      Apply the move-up Procedure (Algorithm 7)

    end

  else

    Go to next level: $l = l + 1$

  end

end

Result: the best solution $BestSol$ and its coordinates $bestPosition$

Algorithm 6: Detailed FDA Algorithm

Input: Deep of the fractal decomposition: $k = 5$ and the tolerance threshold: $\omega_{\min} = 10^{-20}$

Input: Coefficient step-size: $\lambda = 0.5$, inflation coefficient: $\alpha = 1.75$ and dimension of the problem: D

Initialization of the current hypersphere H and the number of function evaluations $NBEval = 0$

Initialize the center $\vec{C}$ at the center of the search space.

Evaluate the objective function of the center $\vec{C}$, initialize the best solution *BestSol* with the resulting value and the best position *bestPosition* with $\vec{C}$; NBEval = NBEval + 1

Calculate the radius r using eq. 2.2; Initialize the level variable: $l = 1$

while *Stopping criteria are not reached* **do**
- Decompose the current hypersphere H using the Fractal procedure using expression (2.3)
- **foreach** $2 \times D$ *l-level hypersphere* **do**
  - Compute g_1, g_2 and g_c using the expression (2.35) and evaluate the quality of the HyperSphere q, using the expression (2.34)
  - NBEval = NBEval + 3
- **end**
- Sort the $2 \times D$ hyperspheres at the current l-level
- Replace the current hypersphere H by the first of the sorted hyperspheres at the current level
- **if** $l == k$ **then**
  - **for** *Each* $2 \times D$ *of* k^{th}*-level hypersphers* **do**
    - Set the solution $\vec{x}_C$ to the center $\vec{C}$ of the current hypersphere H
    - Evaluate the objective function of the solution $\vec{x}_C$
    - $NBEval = NBEval + 1$
    - Set the step size ω, to the radius of the current hypersphere H
    - **while** $\omega \geq \omega_{min}$ **do**
      - **for** *Each dimension* $i = 1, \ldots, D$ **do**
        - $\vec{x}_L = \vec{x}_C - \omega \times \vec{e}_i$
        - $\vec{x}_R = \vec{x}_C + \omega \times \vec{e}_i$
        - Evaluate the fitness of $\vec{x}_L$ and $\vec{x}_R$
        - NBEval = NBEval + 2
        - Update $\vec{x}_C$ by the best solution among $\{\vec{x}_C, \vec{x}_L, \vec{x}_R\}$.
      - **end**
      - **if** *No improvement of the fitness* $\vec{x}_C$ **then**
        - Decrease the step size ω: $\omega = \omega \times \lambda$.
      - **end**
    - **end**
    - **if** *The fitness of* $\vec{x}_C$ *is less than BestSol* **then**
      - Update the best solution *BestSol* with the fitness of $\vec{x}_C$ and *bestPosition* with $\vec{x}_C$
    - **end**
  - **end**
  - **if** *stopping criterion is not reached* **then**
    - Set l Current level, N number of explored hypershperes at level $l - 1$ and D dimension of the problem
    - **while** $N == 2 \times D$ **do**
      - $l = l - 1$
      - Update N to number of explored Hyperspheres at level l
    - **end**
    - **if** $l == 1$ **then**
      - All hyperspheres have been explored
      - Stopping criterion satisfied
    - **else**
      - Update the position of the current hypersphere by the next unexplored hypersphere at the current level l
    - **end**
  - **end**
- **else**
  - Go to next level: $l = l + 1$
- **end**

end

Result: The best solution *BestSol* and its coordinates *bestPosition*

with:

$$g_1 = \frac{f(\vec{s_1})}{\|\vec{s_1} - BSF\ \|}, g_2 = \frac{f(\vec{s_2})}{\|\vec{s_2} - BSF\ \|}\ and\ g_c = \frac{f(\vec{C^l})}{\left\|\vec{C^l} - BSF\right\|} \tag{2.35}$$

where:

$$\vec{s_1} = \vec{C^l} + \alpha\frac{r_l}{\sqrt{D}} \times \vec{e}_d, \quad \text{for} \quad d = 1, 2, ..., D \tag{2.36}$$

$$\vec{s_2} = \vec{C^l} - \alpha\frac{r_l}{\sqrt{D}} \times \vec{e}_d, \quad \text{for} \quad d = 1, 2, ..., D \tag{2.37}$$

2.4.2 Multilevel search strategy

At each level, hyperspheres that have not yet been decomposed are stored in a list, sorted by their quality score, evaluated during the exploration strategy detailed in the Section 2.4.1.

In the case where all the spheres at a level have been explored without reaching the stopping criterion, the next hypersphere in the upper level's $(l-1)$ list, is then chosen to be decomposed. If all the hyperspheres in the upper level have been explored, then, a move to $l-2$ is performed and so on, until exploring the whole search space or the stopping criterion is satisfied as detailed in the Algorithm 7.

Algorithm 7: The move-up procedure

Input: l Current level
Input: N number of explored hypershperes at level $l-1$
Input: D dimension of the problem
while $N == 2 \times D$ **do**
    $l = l - 1$
    Update N to number of explored Hyperspheres at level l
end
if $l == 1$ **then**
    All hyperspheres have been explored
    Stopping criterion satisfied
else
    Update the position of the current hypersphere by the next unexplored hypersphere at the current level l
end

2.4.3 Intensive Local Search (ILS)

Different local searches or even metaheuristics can be used at this level. As in this work, our goal is to design a deterministic, simple and efficient metaheuristic adapted to the large scale problems, a simple local search was considered.
In this local search, two candidate solutions are evaluated per dimension of the search space, denoted by $\vec{x}^{s_1}$ and $\vec{x}^{s_2}$. They stand in opposite directions from the current solution $\vec{x}^{s}$ along an axis of the search space at equal distance α, also called step size:

$$\vec{x}^{s_1} = \vec{x}^{s} + \omega \times \vec{e}_i \tag{2.38}$$

$$\vec{x}^{s_2} = \vec{x}^{s} - \omega \times \vec{e}_i \tag{2.39}$$

where $\vec{e}_i$ is the unit vector which the i^{th} element is set to 1 and the other elements to 0. The step size ω is set to the radius of the current hypersphere being exploited.

Then, the best solution among $\vec{x}^{s}$, $\vec{x}^{s_1}$ and $\vec{x}^{s_2}$ is selected to be the next current solution $\vec{x}^{s}$. The adaptation of the step size ω is performed through the procedure described in Algorithm 8. Depending on the situation, the step size is adapted using the following rules:

- if there is any better candidate solution found in the neighborhood of $\vec{x}^{s}$, then, ω is halved,
- the step size is decreased until a given value as the tolerance or the precision need.

This heuristic is similar to the well-known Hooke-Jeeves Pattern Search method [Hooke & Jeeves, 1961].

2.5 Results and discussions

In this section, the proposed algorithm (FDA) is analyzed and its performance is exposed in the following problems.

2.5.1 Benchmark Functions

The experimental tests were performed on 19 functions (F_1-F_{19}) for large-scale continuous optimization taken from the special issue of soft computing on scalability of evolutionary algorithms (SOCO 2011). The first six functions F_1-F_6 are described in

Algorithm 8: ILS procedure

Input: $\omega_{\min} = 10^{-20}$. //precision or tolerance error
Input: Coefficient step-size: $\lambda = 0.5$
Input: D // the dimension of the problem
Input: Number of fonction evaluations $NBEval$
Input: The current best solution $BestSol$ and its coordinates $bestPosition$
Set the solution $\vec{x}_C$ to the center $\vec{C}$ of the current hypersphere H
Evaluate the objective function of the solution $\vec{x}_C$
NBEval = NBEval + 1
Set the step size to the radius of the current hypersphere H
while $\omega \geq \omega_{min}$ **do**
  for *Each dimension* $i = 1, \ldots, D$ **do**
    $\vec{x}_L = \vec{x}_C - \omega \times \vec{e}_i$
    $\vec{x}_R = \vec{x}_C + \omega \times \vec{e}_i$
    Evaluate the fitness of $\vec{x}_L$ and $\vec{x}_R$
    NBEval = NBEval + 2
    Update $\vec{x}_C$ by the best solution among $\{\vec{x}_C, \vec{x}_L, \vec{x}_R\}$.
  end
  if *No improvement of the fitness* $\vec{x}_C$ **then**
    Decrease the step size ω: $\omega = \omega \times \lambda$.
  end
end
if *The fitness of* $\vec{x}_C$ *is less than BestSol* **then**
  Update the best solution $BestSol$ with the fitness of $\vec{x}_C$ and $bestPosition$ with $\vec{x}_C$
end
Output: $\vec{x}_C$

[Tang et al., 2007], whereas the function F_9 is detailed in [Whitley et al., 1995]. While functions F_{12}-F_{19} are obtained by hybridizing a non-separable function F_{ns} with other functions from the benchmark. This hybridization consists of splitting via the parameter m_{ns} that defines the ratio of variables that are evaluated by F_{ns}. All these functions are exposed in Table A.1 and their properties are detailed in Tables A.2-A.3. Tests were done for the set of dimensions $D = 50,\ 100,\ 200,\ 500$ *and* 1000 and the stopping criterion was defined by the maximum number of function evaluations (FEs) set to $5000 \times D$. For the comparison, the stochastic based algorithms were run 25 times for each function of the benchmark.

2.5.2 Parameters Settings

Parameters of FDA are summarized bellow and were fitted empirically:

- The fractal depth (k) is set to 5. This parameter corresponds to the number of decomposition levels to reach.
- The stopping criterion for the ILS is related to a tolerance threshold (ω_{min}) set to $1 \times e^{-20}$. This chosen value is problem dependent. For these experimentations, this value is equal to precision of the shift values in the shifted functions F_1-F_6.
- The decrease coefficient of the step-size λ, is set to the standard value 0.5.
- The inflation coefficient α, was set to 1.75.

2.5.3 Sensitivity analysis of FDA

In this subsection, the sensitivity analysis of FDA against its parameters is presented. The fractal depth k is the parameter that has an impact on the performance of FDA. The rest of the parameters can be set to values presented in Section 2.5.2.

In these experimentations, the parameter k was varied, while, other parameters were set at their suited values. Table A.4 summarizes results on only some functions, because FDA reaches the global optimum for the rest. As it can be seen from obtained results, this parameter is important and the performance of FDA varies against its value. However, for the considered set of functions, the value 5 seems to be the most suited.

2.5.4 Complexity Analysis

The proposed approach includes three distinct parts: the first is the fractal decomposition process; the second consists of the quality's evaluation of the hypersphere, while, the third is the application of ILS.

Their asymptotic complexities are presented in Table A.5, respectively, where D represents the problem dimension, r the radius of the current hypersphere and ω_{min} the ILS tolerance threshold.

Using Table A.5, the complexity of the FDA is given by (2.40). Hence, the asymptotic complexity shows that FDA has a logarithmic complexity depending on the fractal depth parameter: $\mathcal{O}_{FDA}(\log_k(D))$:

$$\mathcal{O}(\log_k(D) + 1 + \log_2(r/\omega_{min})) = \mathcal{O}(\log_k(D)) \tag{2.40}$$

Besides, the FDA memory complexity is $\Theta(D)$.

2.5.5 FDA Results

The reported results of the Fractal Decomposition based Algorithm are the error values $f(x) - f(x^*)$ obtained for dimensions $D = 50, 100, 200, 500$ and 1000. These results are presented in Tables A.6-A.8 through statistical measures. In our case, the mean and the standard deviation are sufficient to describe the behavior of the FDA for each test function knowing that all the obtained standard deviations are equal to 0. Besides, all average errors below 10^{-14} are considered equal to 0 as suggested in [Lozano et al., 2011] where the benchmark is detailed.

As it was expected, the proposed approach seems to have difficulties in solving the Shifted Rosenbrock's function F_3, and the hybrid composition functions involving F_3, F_{13} and F_{17} because our choice of ILS is more suited for separable and weakly separable problems. One can use another heuristic or metaheuristic rather than ILS. However, the FDA was able to reach the global optimum for 14 out of the 19 tested functions and that, for all the dimensions presented.

On the other hand, the fact that the standard deviations are always equal to 0 denotes that the algorithm reaches always the same optimum.

2.5.6 Analysis of FDA's behavior

This section focuses on illustrating the way in which FDA behaves in terms of the number of spheres visited, fitness convergence and function evaluation consumption. The aim is to understand the behaviour of our proposed algorithm on the three main functions types: 1) Separable function, 2) Weakly separable function and 3) Non-Separable functions.

To illustrate its behavior, three functions (one of each type) taken from the benchmark SOCO 2011 were considered. The Shifted Rosenbrock's Function (F_3) and the Shifted Rastrigin's Function (F_4) are defined in the Table A.1. For the weakly separable function, the composite function F_{16} has been selected (defined in Table A.3). For each function, FDA's behaviour is presented for dimensions $D = 50$ and $D = 1000$ (being the smaller and the bigger on the benchmark).

Table A.9 and Figure 2.3 show the number of evaluation consumed to find the best solution possible for both mentioned dimensions. For the separable function (F_3) all the function evaluations allowed are consumed without reaching the global optimum in both dimensions $D = 50$ and $D = 1000$. As pointed out, this is due to the intrinsic nature of the ILS. However, for the two other functions, weakly non-separable and

separable functions, the global optimum is reached. For F_3, in dimension D = 50, the optimum is reached in 7803 evaluations over 25000 allowed, representing around 31% of the stopping criteria. In D = 1000, the optimum is reached in 160002 over 500000 allowed, representing around 32% of the allowed function evaluations. This highlights the performance stability and scalability of FDA. In the case of the weakly separable function, F_{16}, FDA reached the optimum after 12802 function evaluations, in dimension D = 50, (over the 25000 allowed), representing 51% of the permitted evaluation and 268002 evaluations over 500000 in D = 1000, around 54%. This confirms the stability and scalability of FDA.

Furthermore, the total number of hyperspheres visited, meaning all hyperspheres focused by FDA at all l levels, in both exploration and intensification phases is given in Table A.10 and illustrated in Figure 2.4. In each function, the number of visited spheres is the same for both dimensions ($D = 50$ and $D = 1000$) showing the stability and scalability of the algorithm and our proposed decomposition approach regardless the dimension.

Finally, to illustrate FDA's behaviour, the fitness convergence is shown in Figure 2.5 illustrating the fitness over the number of function evaluations. A log function has been applied on both axis. Once again, for each function, the behaviour is similar across dimensions. In addition, in this figure we can observe that the slope drops suddenly and significantly to reach the moment when the best solution is found, being the optimum for F_3 and F_4. This sudden change corresponds to the moment when ILS is triggered and as explained, in the case of non-separable function (F_3 in our case) the curve stabilised without improving significantly until stopping criteria is reached.

In summary, FDA has a stable and scalable behaviour across all dimensions in the case of separable and weakly separable functions, keeping the number of visited spheres constant and the percentage of allowed function evaluations constant as well.

2.5.7 Comparison with competing algorithms

In this section, a comparison of our proposed FDA algorithm is conducted with other optimization algorithms from the literature. In first, a comparison with the related algorithm: DIRECT is performed. Then, FDA is compared with other competing metaheuristics from the literature, theirs results on SOCO 2011 were taken from the corresponding papers.

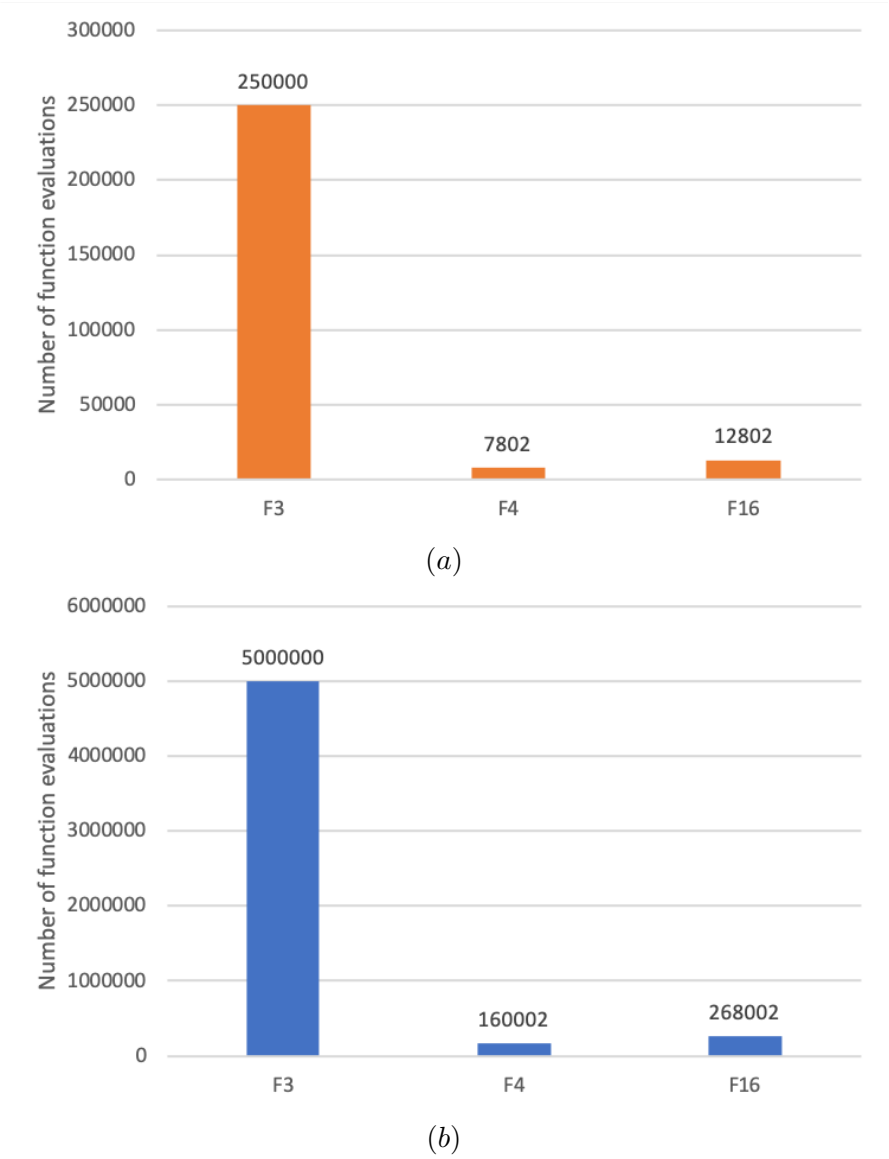


FIGURE 2.3: Illustration number of evaluations to find the best solution for F_3, F_4 and F_{16}. (a) $D = 50$, (b) $D = 1000$.

2.5.7.1 Comparison with DIviding RECTangles (DIRECT)

As pointed out in the literature, DIviding RECTangles (DIRECT) is an algorithm that decomposes the search domain to find the global optimum. Being one of the most popular in the Multi-Scale Optimization (MSO) category [Al-Dujaili et al., 2016a], we have decided to compare its results with FDA. However, as its number of expansions grows quadratically with regards to the problem dimension N, we restricted the comparison to functions F_1 to F_6 (Table A.1), and to the dimensions $D = 50$ and $D = 100$.

As mentioned in the literature, DIRECT does not perform well in dimension $D > 10$. This is confirmed by the results shown in Table A.11 as FDA outperforms DIRECT on

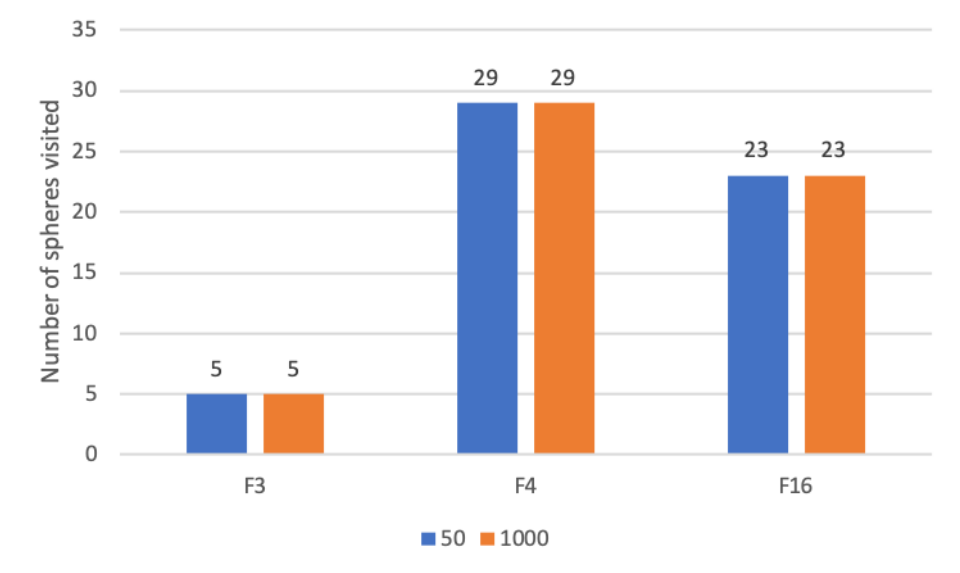


FIGURE 2.4: Illustration the number of hyperspheres visited for F_3, F_4 and F_{16} for dimensions $D = 50$ and $D = 1000$.

both dimensions and on all functions.

2.5.7.2 FDA comparison with SOCO 2011 Participants

As mentioned the first seven metaheuristics considered are presented. However, algorithms based on the hybridization of multiple metaheuristics were excluded because we consider them as a separate class of metaheuristics. Then, the considered algorithms are:

- Differential Evolution Algorithm [Storn & Price, 1995] which uses the exponential crossover ($DE/rand/1/exp$).
- Real-coded Genetic Algorithm (CHC) [Eshelman & Schaffer, 1992].
- MA-SSW-Chains: Memetic algorithm based on local search chains for large-scale continuous Optimization Problems [Molina et al., 2011]. In addition to classical memetic algorithm, this version consists of applying a local search, to the last used configuration.
- The first local search method issued from the Multiple Trajectory Search for large-scale optimization [Tseng & Chen, 2008] presented as MTS-LS1 as in [LaTorre et al., 2011]. The algorithm was fitted using the suited values suggested by authors of the original paper [Tseng & Chen, 2008].

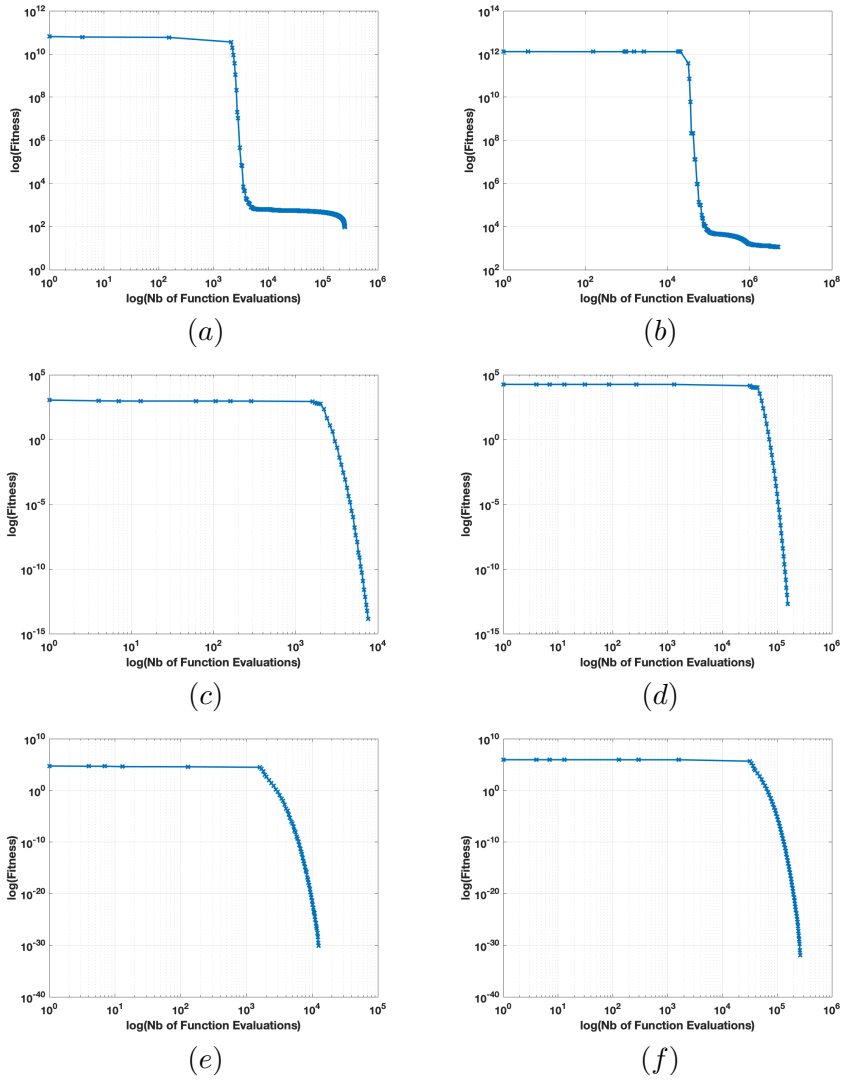


FIGURE 2.5: Diagrams illustrating the log of the fitness over the log of number of function evaluations. (a) F_3 and $D = 50$, (b) F_3 and $D = 1000$ (c) F_4 and $D = 50$ (d) F_4 and $D = 1000$ (e) F_{16} and $D = 50$, (f) F_{16} and $D = 1000$.

- Self-adaptive Differential Evolution (SaDE) [Qin & Suganthan, 2005] that uses a learning procedure to generate trial vectors strategies with their associated parameters.

- Multi-population Differential Evolution with balanced ensemble of mutation strategies for large-scale optimization (mDE-bES) [Ali et al., 2015]. This algorithm consists of dividing the population into independent subpopulations using different mutation operators for each one and updating the strategies during the search.

- Self-adaptive differential evolution algorithm using population size reduction and three strategies (jDElscop) [Brest & Maučec, 2011]. jDElscop employs three differential evolution (DE) strategies, a newly proposed population size reduction

mechanism, and a mechanism for changing the sign of F control parameter.

Table A.19-A.23 illustrates the average error obtained by FDA on each algorithm presented above. Table A.16 indicates the number of times each algorithm reaches the global optimum for all the benchmark's dimensions, respectively $D = 50$, $D = 100$, $D = 200$, $D = 500$ and $D = 1000$. To perform the performance comparison, the eight algorithms are ranked using the Friedman ranking sum test presented in Tables A.13 and illustrated on Figure 2.6. In other terms, the average relative rank is computed for each algorithm according to its mean performance for each function and the average ranking computed through all the functions is then reported. It can be observed that our algorithm is ranked first for all dimensions. It is also illustrated in Figures 2.8-2.12 which show the boxplots for the distribution of the average ranks for each algorithm on all functions. In those plots, the circle highlights the outliers, meaning the functions where the algorithm performs surprisingly good or bad. In our case FDA performs surprisingly bad on the function F_{13} on dimension $D = 50$. It is however clear that our work shows better and stable performance among all dimensions on all functions.

To confirm this performance, we have conducted a Wilcoxon pairwise test to FDA and each algorithm presented. The p-values given by the Wilcoxon test have been adjusted using the Holm procedure [LaTorre et al., 2014] to control the familywise error rate. Table A.13 presents the Friedman Rank Sum score for each algorithm and shows that FDA is ranked first in all dimensions. Table A.14 and Table A.15 show the resulting p-values (raw and adjusted) of the Wilcoxon test. Thus, algorithms with a p-value <0.05 are statistically outperformed by our proposed work. Looking at the p-values, FDA statistically outperformed MA-SSW-Chains, CHC, DE, MTS-LS1 and SaDE in all dimensions. Finally, to support our work performance, as shown in Table A.16, FDA solved, on average, 14 problems out of 19, and is ranked 1^{st} where both mdE-bES and jDElscop solve, in rounded-average, 9 problems and are respectively ranked 2^{nd} and 3^{rd}. We can conclude that FDA performs well for the considered benchmark problems scalability-wise and is stable among all dimensions.

Regarding the complexity, Table A.12 summarizes different complexities of algorithms. It can be noticed that FDA has the lowest complexity. Moreover, the rest of the algorithms in Table A.12 have polynomial complexities, while, FDA has a logarithmic complexity. Hence, the theoretical analysis of the proposed approach in addition to the obtained experimental results shows that the FDA can be an efficient alternative to solve large-scale problems.

2.5.7.3 Comparison with recent metaheuristics

In addition to the study conducted in the previous section, a similar experimentation has been performed to compare the performance of FDA with recent metaheuristics. These algorithms were inspired by a comprehensive comparison of large scale global optimizers [LaTorre et al., 2014] and other algorithms taken from the literature and are described in the following. While in the previous section, hybrid algorithms using multiple metaheuristics were excluded as part of a different class, we have included some examples in that section as they are state-of-the-art approaches.

- Multiple Offspring Sampling (MSO) based dynamic memetic differential evolution algorithm for continuous optimization referred to as MSO-SOCO2011. A hybrid version of MSO combining a differential evolution (DE) algorithm and the first one of the local searches of the MTS algorithm [LaTorre et al., 2011].

- Multiple Offspring Sampling in Large Scale Global Optimization (MSO-CEC2012). Another hybrid MSO-based metaheuristic combining the first one of the local searches of the MTS [LaTorre et al., 2011] (also used in MSO-SOCO2011) and the Solis and Wets heuristic [Solis & Wets, 1981]. Originally applied to the CEC 2005 and CEC 2012, the results for the SOCO-2011 have been found in the literature [LaTorre et al., 2014].

- Large Scale Global Optimization: experimental results with MOS-based hybrid algorithms [LaTorre et al., 2013] (MOS-CEC2013). Yet another MSO-based approach. This version also combines the power of the MTS [LaTorre et al., 2011] with the Solis Wets' algorithm [Solis & Wets, 1981] but innovates in integrating a population-based search Genetic Algorithm (GA).

- Two-stage based ensemble optimization for Large-Scale Global Optimization referred as $2S - Ensemble$ in [LaTorre et al., 2011] and presented in the original paper [Wang et al., 2013]. The search procedure is divided into two phases: 1) the global shrinking focusing on finding a promising area as fast as possible using an EDA based-on mixed Gaussian and Cauchy models (MUEDA) [Wang & Li, 2009] and 2) exploring the selected area using a co-evolution-based algorithm.

- IACO$_{\mathbb{R}}$-Hybrid is a hybridisation method for the exploration phase, based on an Incremental Ant Colony Framework [Liao et al., 2011] (IACO$_{\mathbb{R}}$) and combining the Multi-Trajectory Local Search (Mtsls1) algorithm and Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [Nocedal & Wright, 2006].

The same experimentations, as in the previous section, were performed to compare FDA with these metaheuristics, Table A.24 shows the average errors obtained by each algorithm based on 19 functions of the SOCO 2011 benchmark with a focus on the dimension D = 50 as results were available in the original papers and in the literature for all algorithms.

Table A.17 shows the ranks using the Friedman sum test where FDA is ranked second. This is also illustrated in the Figure 2.7. The Figure 2.13 illustrates as a boxplot, the distribution of the ranks of all algorithms. As mentioned in the previous section, circles represent the outliers. In that case, FDA performs surprisingly bad in 3 functions, the first, third and sixth causing the rank to drop second. Overall FDA performance is stable on all other 16 functions.

Table A.18 presents the p-values, adjusted using the Holm procedure, resulting from a Wilcoxon pairwise test to FDA and allows us to highlight the fact that FDA is statistically more efficient than MOS-CEC2013, MOS-CEC2012 and 2S-Ensemble. While MOS-SOCO2011 and IACO$_{\mathbb{R}}$-Hybrid appear to achieve better performance than our work by obtaining respectively similar and better ranks, the adjusted p-values from the Wilcoxon test lead us to ensure that no statistical differences can be found to confirm that impression. In addition, Table A.25 shows that FDA solved the same number of problems as the other two most performant algorithms.

Finally, it is crucial to emphasize the fact that, as mentioned earlier, FDA uses a deterministic approach with a single solution, hence and benefit from neither a population solution nor a stochastic approach, unlike the other state-of-the-art algorithms. Failing to find significant differences with FDA highlights the power of FDA and its potential margin for improvement and, as detailed in the future work section, the FDA approach will lead to great results by incorporating elements which can be found in state-of-the-art algorithms.

2.6 Conclusion

In conclusion, a new deterministic metaheuristic to solve large-scale optimization problems has been proposed. The approach includes a *divide and conquer* mechanism to explore the search space. Indeed, the geometric fractal decomposition uses the hypersphere as a geometrical form to represent the search space and its subregions to be visited. Then, a heuristic with a minimum cost in terms of complexity is applied to lead the search to a smaller promising region allowing the ILS to intensify the search to find the best solution in a reduced area at the last level. The Fractal Decomposition based

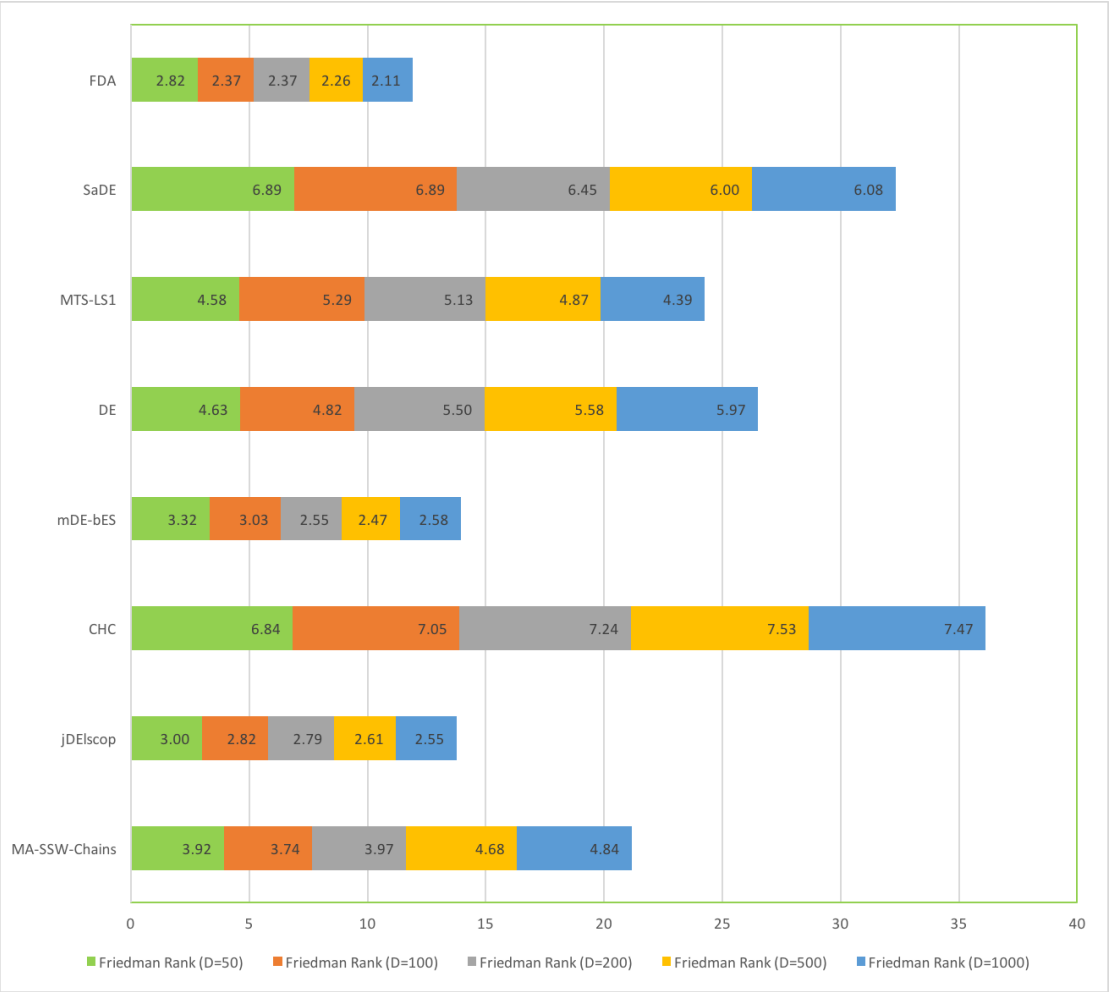


FIGURE 2.6: Illustration of the ranking of the algorithms at dimensions $D = 50$, $D = 100$, $D = 200$, $D = 500$ and $D = 1000$.

Algorithm was tested on a set of test functions issued from the benchmark provided for the soft computing special issue on the scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems. The obtained results show the efficiency of the proposed approach and the comparisons with other state-of-the-art algorithms taken from the literature prove that its performance is very competitive for all considered dimensions.

As it was pointed out, the procedure used in ILS does not allow solving of problems that are fully non-separable structure. In near future, our work consists of proposing new heuristics at ILS level to deal with these problems. In the next chapter, we introduce PFDA a parallelized implementation of FDA on multi-threaded environments and MA-FDA a parallelized implementation on multi-nodes environments.

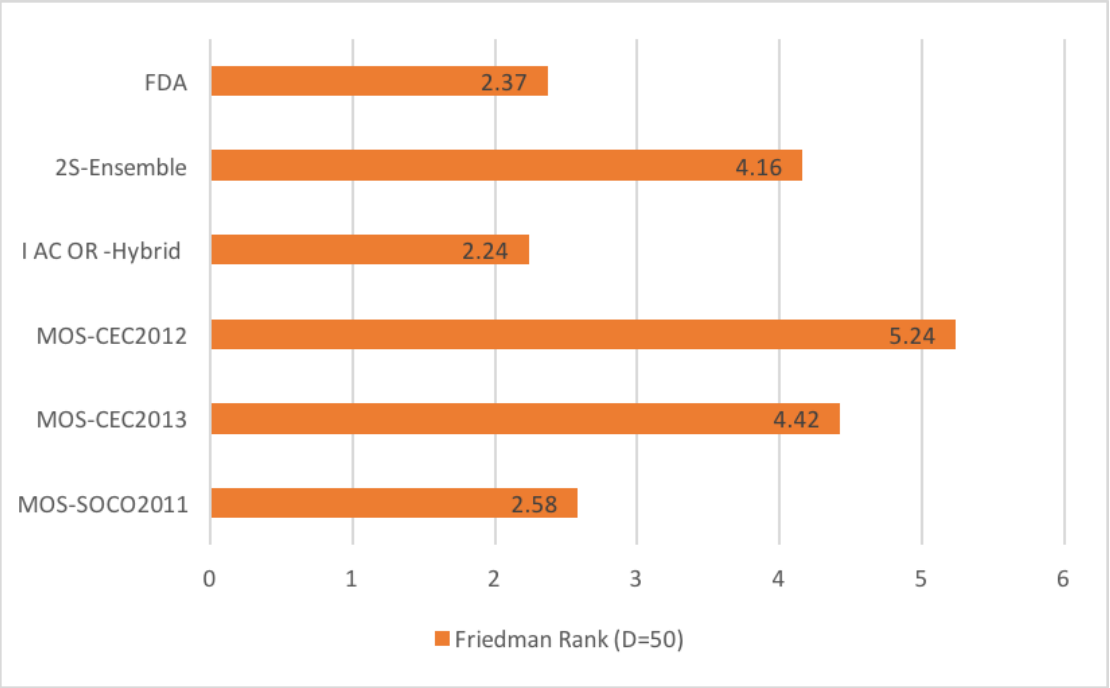


FIGURE 2.7: Illustration of the ranking of the other metaheuristics algorithms at dimension $D = 50$.

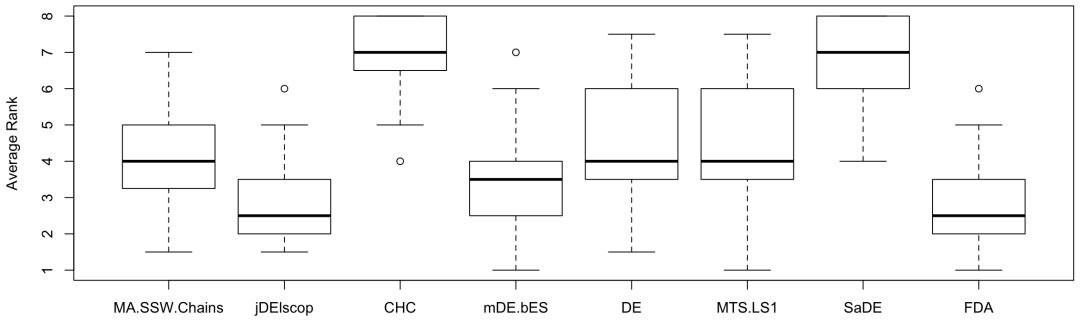


FIGURE 2.8: Boxplots for the distribution of the average ranks for each algorithm on $D = 50$. Circles represent outliers as defined earlier.

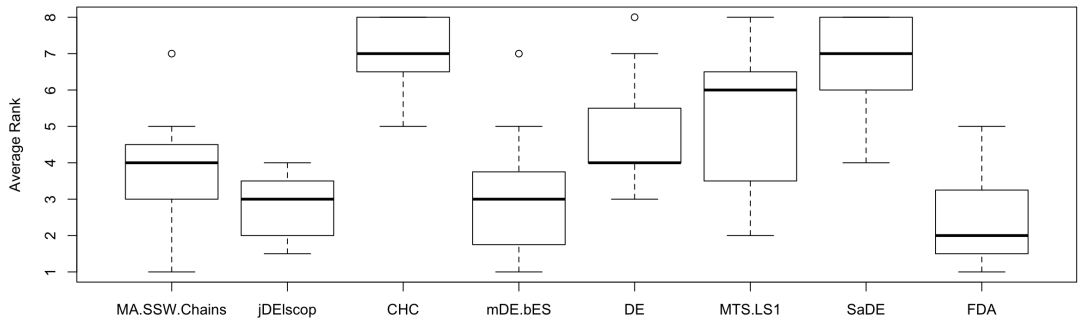


FIGURE 2.9: Boxplots for the distribution of the average ranks for each algorithm on $D = 100$. Circles represent outliers as defined earlier.

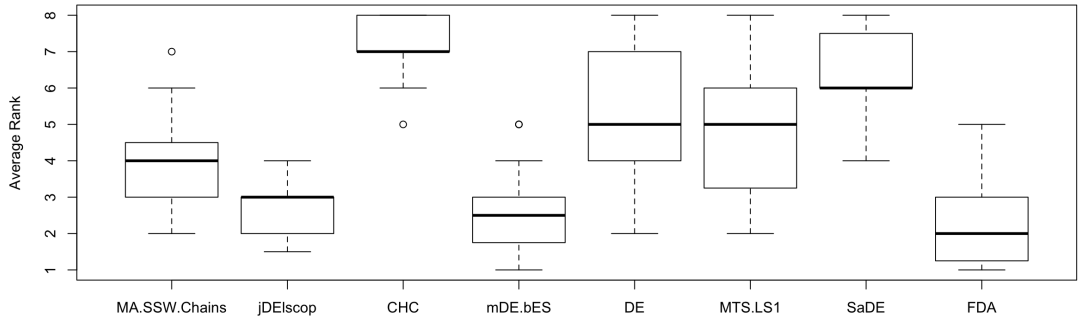


FIGURE 2.10: Boxplots for the distribution of the average ranks for each algorithm on $D = 200$. Circles represent outliers as defined earlier.

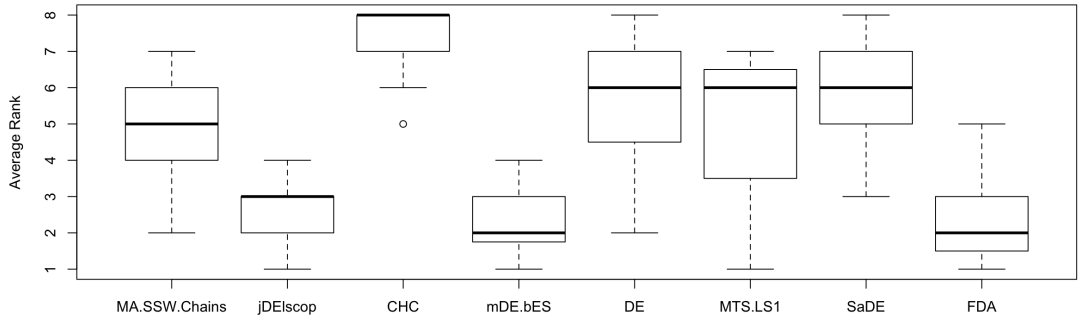


FIGURE 2.11: Boxplots for the distribution of the average ranks for each algorithm on $D = 500$. Circles represent outliers as defined earlier.

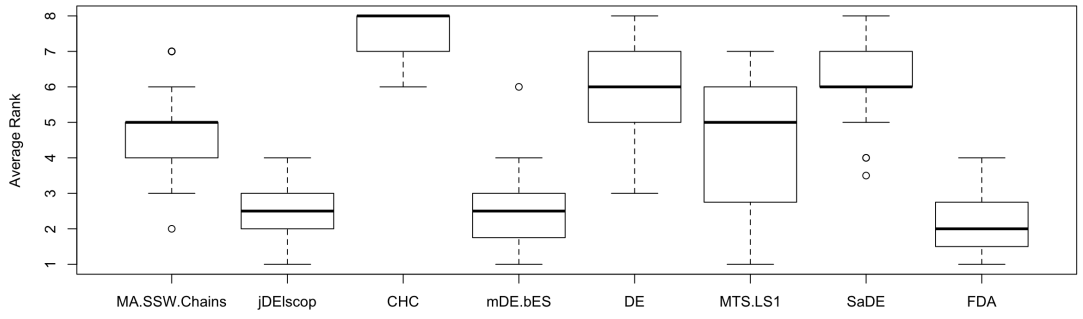


FIGURE 2.12: Boxplots for the distribution of the average ranks for each algorithm on $D = 1000$. Circles represent outliers as defined earlier.

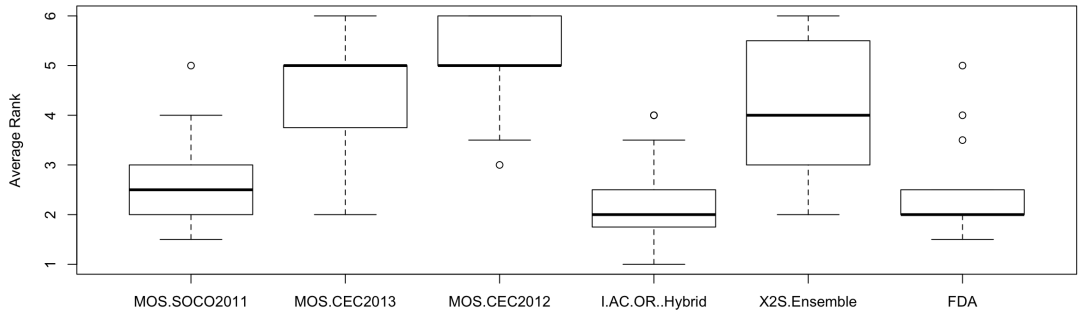


FIGURE 2.13: Boxplots for the distribution of the average ranks for each algorithm at dimensions $D = 50$. Circles represent the outliers as explained earlier.

Chapter 3

Parallel fractal decomposition based algorithm for large scale continuous optimization problems

3.1 Introduction

FDA would easily benefit from multi-threaded and multi-node environments. Indeed, recent architectures are composed of important machines containing many threads and/or cluster of smaller machines.

In this chapter, we present two different modification of our algorithms. One benefits from multi-threaded environments while the other from multi-node environments. Both approaches are designed to leverage current available IT resources such as the ones provided by new cloud infrastructures. The multi-threaded approach is called, "Parallel fractal decomposition based algorithm" (PFDA) [Nakib et al., 2018] and the multi-node is called "Multi-Agents Fractal decomposition based algorithm" (MA-FDA).

In its original version, FDA was running on a mono-threaded environment and therefore its computational time increases significantly when the problems' dimension increases. The motivation of the current work was to address this issue. Reducing the execution time, solving big optimization problems (problems with dimensions higher than 1000), and maintaining the original precisions. In this chapter, a parallelized version of FDA, called PFDA, running on a multi-threaded environment using the framework OpenMP[1] and following the Fork/Join model is proposed. This approach is motivated by the fact

[1]OpenMP 4.5 is used in this thesis

that each thread can explore and exploit hyperspheres simultaneously. The aim is to significantly improve its running time with a focus on large scale optimization problems.

While PFDA has been developed to leverage large machines with many threads available, modern IT infrastructures may involve clusters of machines with a limited number of threads. To benefit from distributed multi-node environments FDA has been adapted accordingly. This Chapter also presents MA-FDA, the adapted version of FDA for multi-node environments.

The rest of this paper is organized as follow: In Section 3.2 an analysis of the mono-threaded FDA is presented. Section 3.2.1 details our modified approach PFDA. Section 3.2.2 illustrates and discusses the results obtained by the multi-threaded implementation of the FDA algorithm. Section 3.2.3 presents MA-FDA, the multi-node implementation of FDA and its results are discussed in Section 3.2.4. Finally, a conclusion in Section 3.3 ends this chapter.

3.2 Analysis of the mono-thread implementation of FDA

To understand the motivation behind the implementation of PFDA it is important to understand the life cycle of FDA algorithm. Figure 3.1 shows the main life cycle using Unified Modeling Language (UML).

The Figure 3.2 illustrates the four main phases of FDA. In Figure 3.2 (a) represents the first hypersphere (in red) being decomposed into $2 \times D$ child-hyperspheres (CH_i). For more clarity, in this example the dimension is set to $D = 2$. It can be seen on Figure 3.2 (b) that child-hyperspheres are evaluated sequentially. Once the child-hypersphere with the best quality is found (CH_2 colored in red in this case), then, it is also decomposed into $2 \times D$ child-hyperspheres (in Figure 3.2 (c)). When the depth k is reached *(k set to two in our example)*, ILS is triggered on all created child-hyperspheres. In Figure 3.2 (d) one hypersphere is exploited at a time by the heuristic. When all child-hyperspheres of the level k have been exploited, FDA either terminates if the stopping criterion has been reached or backtracks in the search tree and continues.

It is important to highlight the fact that both the exploration (Section 2.4.1) and exploitation (Section 2.4.3) phases handle hyperspheres sequentially and therefore create bottlenecks.

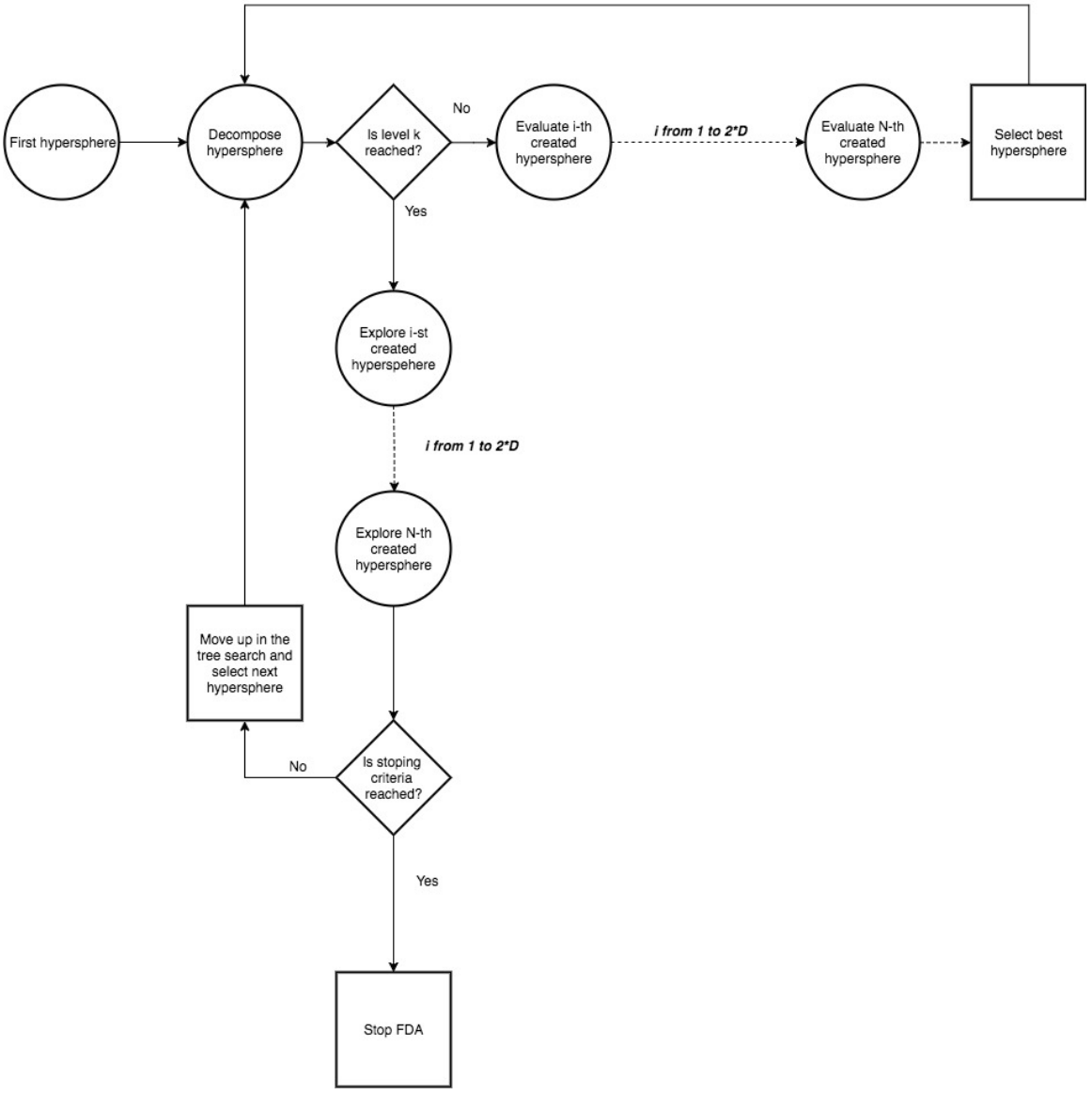
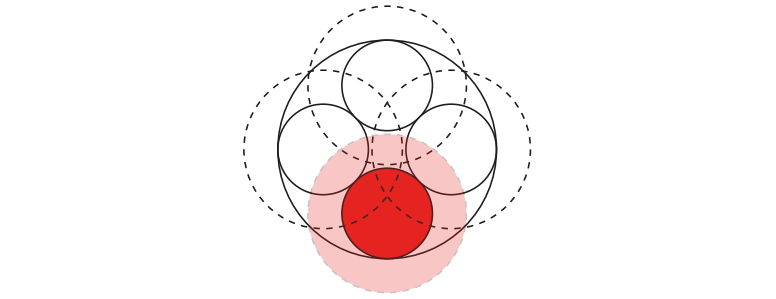


FIGURE 3.1: Illustration of life-cycle of the mono-threaded version of FDA.

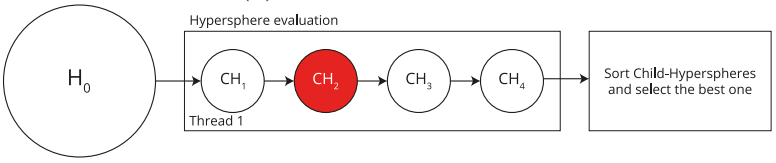
3.2.1 Proposed Multi-threaded Implementation Strategy

Finding a good solution (if the optimum is not known) and within a reasonable time are the two main aspects to be taken into account when designing a metaheuristic. The increase in the complexity of the problem will naturally increase the computation time required for the algorithm to find the desired solution.

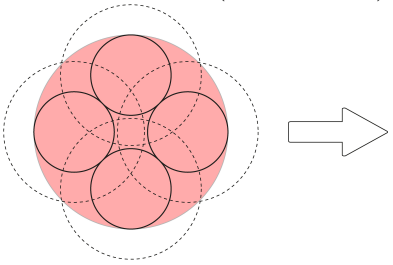
This section describes the proposed Parallel FDA, called PFDA, with OpenMP. When parallelizing, one should aim for achieving a trade-off between improving performance, while minimizing the overhead of the parallelized mechanisms which includes communication, synchronization between threads, memory sharing and simplicity of implementation.



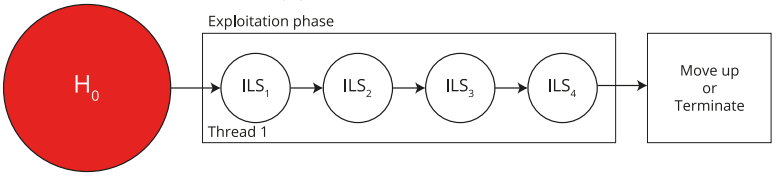
(a) First Decomposition



(b) Evaluation of spheres (Exploration)



(c) Decomposition



(d) Exploitation phase

FIGURE 3.2: Illustration of Exploration phase (a) and (b) and the Exploitation phase (c) and (d) of a $2 \times D$ problem with fractal depth 2 on a single threaded environment. The sphere in red having the highest quality at level 1 (a) and being decomposed (c) for exploitation phase (d).

The idea behind parallelizing FDA was to remove the bottlenecks mentioned earlier, *i.e.* the exploration and exploitation phases. They are also the steps when function evaluations are consumed. Hence, these two phases need to be parallelized.

Using UML, the full life cycle of PFDA is illustrated in Figure 3.3.

The Figure 3.4 illustrates the used strategy based on the previous example. Figures 3.4 (b) and (d) represent respectively the parallelized version of the exploration and exploitation phases, handling hyperspheres simultaneously.

The initialization phase remains on a single thread, the hypersphere is being decomposed and only at this point the exploration phase starts. Instead of evaluating hyperspheres one at a time, from one to N hyperspheres with $N = 2 \times D$, PFDA is able to evaluate

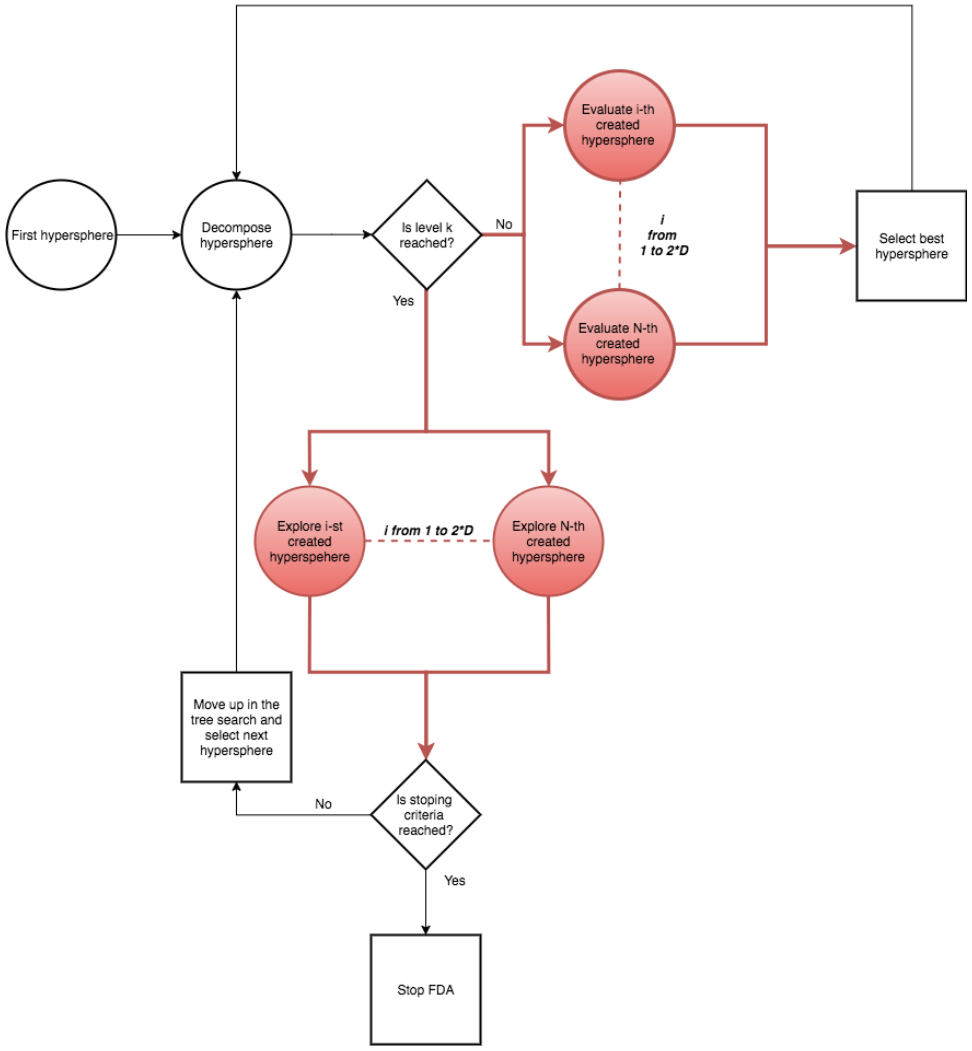
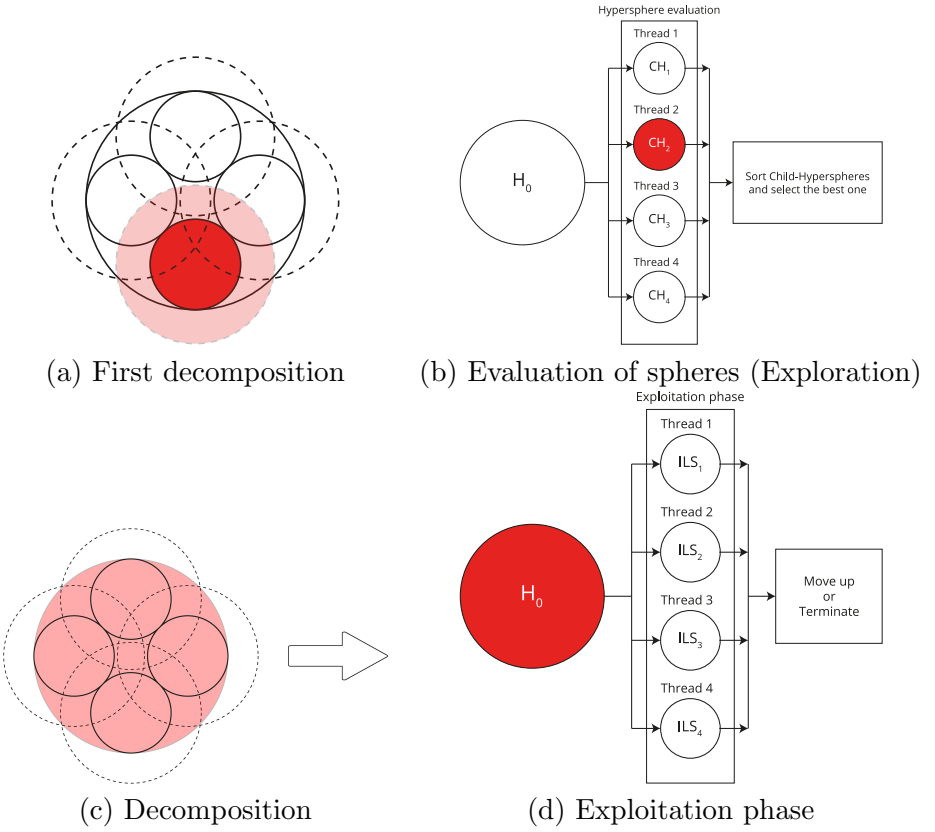


FIGURE 3.3: Illustration of life cycle of PFDA.

hyperspheres in parallel. The algorithm returns to a mono-threaded state and sort all hyperspheres, selecting the best one to be decomposed. This is being repeated until the last level k is reached. At this point, the most promising region is decomposed triggering different instances of ILS. Then, $2 \times D$ generated hyperspheres are exploited in parallel. Once all hyperspheres have been exploited, PFDA terminates if stopping criterion is reached or backtracks in the search tree otherwise.

In other terms, PFDA alternates between mono-threaded and multi-threaded phases which corresponds to the well known $Fork/Join$ model. It is important to notice that the algorithm was designed to be easy to implement.

Regarding the programming environment for implementation, the final choice was OpenMP. Indeed this framework, compatible with the original implementation of our algorithm in C++, is commonly used in the literature for multi-threaded environments and stands



(a) First decomposition (b) Evaluation of spheres (Exploration)

(c) Decomposition (d) Exploitation phase

FIGURE 3.4: Illustration of the proposed strategy. Illustration of Exploration phase (a) and (b) and the Exploitation phase (c) and (d) of a 2D problem with fractal depth 2 on a multi-threaded environment. The child-hypersphere in red having the highest quality at level 1 (a) is being decomposed (c) for exploitation phase (d) which is also ran on a multi-threaded environment.

out in terms of popularity, performance and simplicity of implementation Akhmetova et al. [2017]; Arnautovic et al. [2013]; Di Domenico et al. [2017].

3.2.2 Results and Discussions of PFDA

In this section, the obtained results are presented and analyzed. When adapting an existing metaheuristic it is important to be able to measure the benefits of the improvement. In this case, the main concern is the computational time of the algorithm, this study will focus on the SpeedUp criteria [Alba & Luque, 2006]. This metric is defined by:

$$S = \frac{T_1}{T_n} \tag{3.1}$$

where S represents the SpeedUp, T_1 the execution time of the algorithm on a single thread and T_n, the execution time on n threads.

As shown in Alba & Luque [2006] this is not a valid comparison for non-deterministic algorithms. Originally FDA is deterministic, however, parallelizing the exploration phase adds a stochastic effect. Therefore, the SpeedUp remains suited for evaluating our approach.

3.2.2.1 Performances evaluation

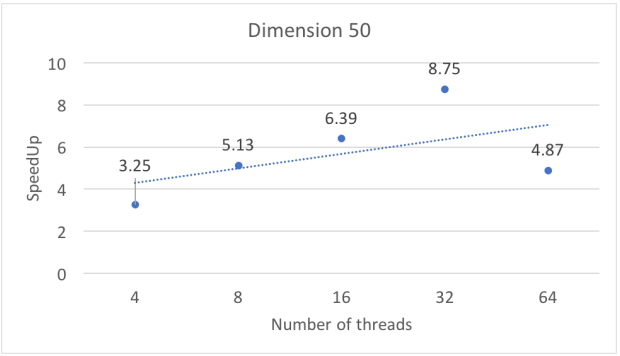
To evaluate the performance of the proposed algorithm on the large-scale continuous optimization benchmark of the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms (SOCO 2011) was considered. This benchmark is composed of six functions from the CEC'2008 special session and competition on large-scale global optimization (Tang et al. [2007]), and other problems generated by hybridizing these functions.

The comparison was performed between the computation time taken by PFDA and that of FDA to solve the benchmark. For the sake of the comparison, the stopping criterion of the benchmark was conserved: the number of functions evaluations set to $5000 \times D$, D *being the dimension of the problem.* In addition, only the dimensions $D = 50$, $D = 100$ and $D = 1000$ have been studied.

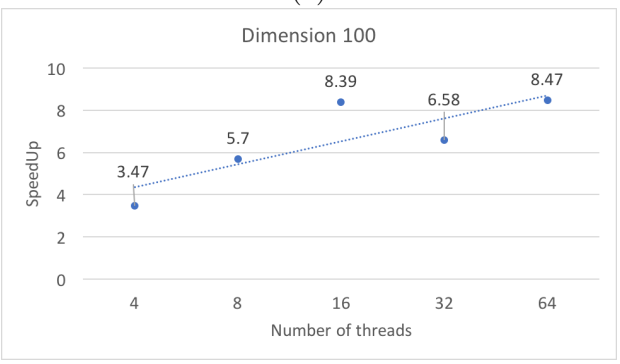
The machine used for experimentations has the following characteristics: a processor Intel Xeon E5-2686 v4 with 256GB of RAM with the technology Intel Turbo Boost Technology. The SpeedUp has been computed on the following number of threads: 4, 8, 16, 32, 64.

In Figure 3.5 variations of the SpeedUp over the number of threads are presented. One can see that the increase in the number of threads allows reducing significantly the running time to reach the stopping criterion. It can also be noticed that for small dimensions, the increase in the number of threads does not automatically decrease the running time. However, for large problems, it is clear that the increase in the number of threads significantly decreases the execution time. The Figure 3.5 illustrates this remark in case of the dimension $D = 1000$, where the SpeedUp is equal to 24.22 with 32 threads.

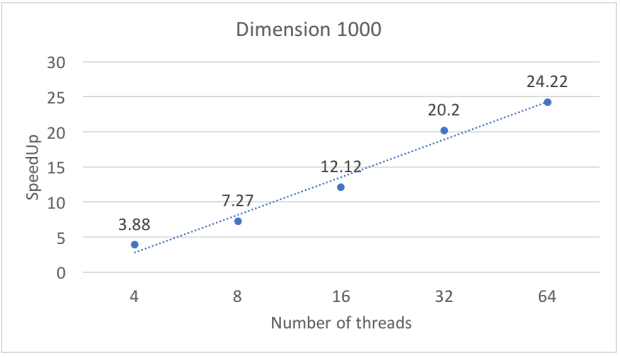
To analyze the performances regarding different kind of problems, a focus was made on first six functions F_1-F_6 Tang et al. [2007], on the dimension $D = 1000$. These six functions represent the different types of problems: separable and non-separable. The best SpeedUp obtained is equal to 27.73 in case of a non-separable problem (Rosenbrock F_3 function). The different SpeedUps obtained for the previous considered problems are



(a)



(b)



(c)

FIGURE 3.5: SpeedUp versus the number of threads for solving the 19 functions (combined). (a) $D = 50$, (b) $D = 100$, (c) $D = 1000$.

presented in Figure 3.6. It can be noticed that in all cases a linear tendency of the increase of the SpeedUp can be observed. This confirms the results illustrated in Figure 3.5.

Regarding the quality of the final solution obtained by the algorithm, the results of both versions (FDA and PFDA) are summarized in Table B.1. It can be noticed from these results that when FDA found the optimum, PFDA also found it. However, the functions where FDA did not find the optimum, results of PFDA are far from the optimum. The quality of the solution, in this case, decreases with the increase of the number of threads. This is due to the stopping criterion being based on the number of function evaluations.

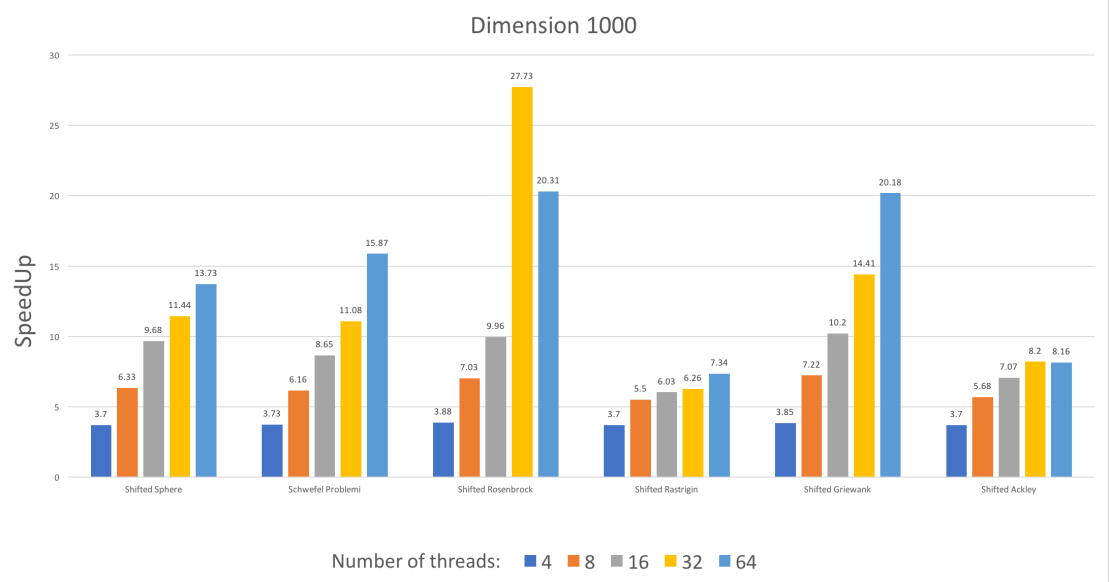


FIGURE 3.6: SpeedUp versus the number of threads concerning the six first functions of the SOCO 2011 benchmark.

Indeed, as $n = 64$, n hyperspheres are being explored at the same time, meaning that functions evaluations are performed in parallel. Hence, PFDA cannot exploit as deep the first hypersphere (supposed to be the most promising one) as FDA, which exploits them one at a time and can, therefore, go deeper in the first hypersphere. For instance, all functions evaluations are consumed in the first hypersphere generated on the last level k in case of the optimization of Rosenbrock problem via FDA.

Hence, FDA intensifies the search in the first hyperspheres more than PFDA can do. Indeed, PFDA exploits n hyperspheres at once.

It is obvious that the parallelized version needs more evaluations of the objective function to reach results similar to those of the single-threaded version. In Figure 3.7, one can see the different SpeedUps obtained by FDA on single thread and PFDA on 64 threads. To analyze the performance in terms of SpeedUp when a target value of the objective function is considered as a stopping criterion. The Figure 3.8 presents obtained results. As it was expected, PFDA reaches similar results in a shorter computational time.

3.2.2.2 Exploring higher dimension

In these experimentations, the goal is to solve big optimization problems via PFDA. The considered problems are the first six functions of SOCO 2011 benchmark, where the dimension is $D = 5000$. The number of thread considered was 64. For the purpose of this study, the stopping criterion will remain at $5000 \times D$.

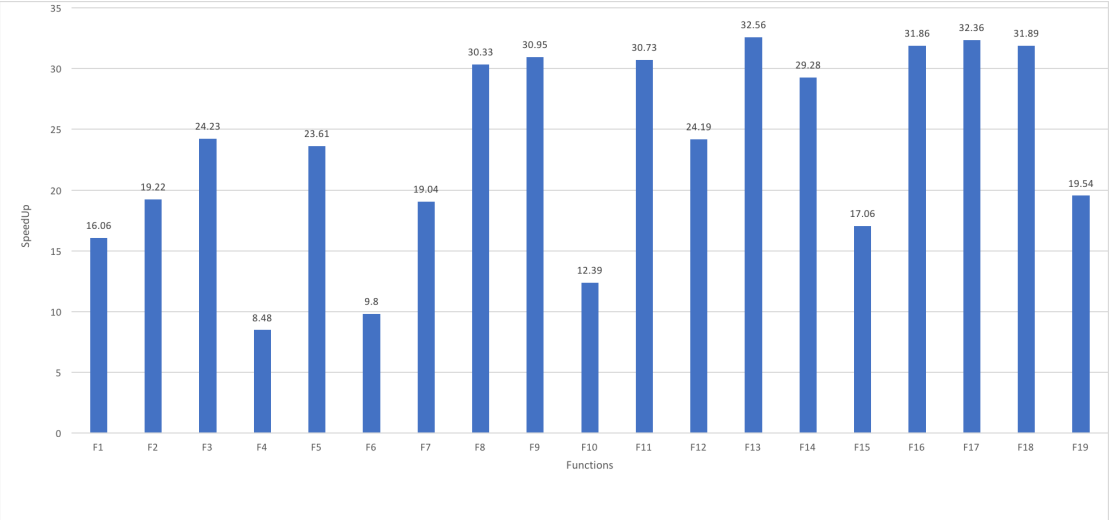


FIGURE 3.7: Obtained SpeedUps on a 64-thread environment with stopping criterion at $20000 \times D$ for both FDA and PFDA.

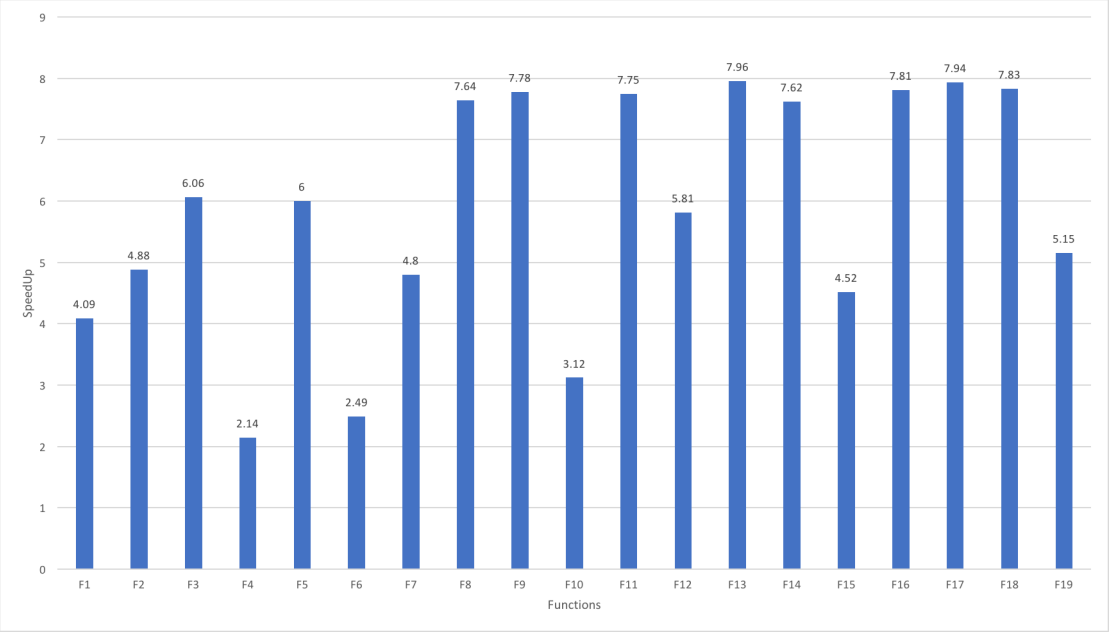


FIGURE 3.8: Obtained SpeedUps of single threaded FDA and 64-thread PDFA when a target value of the objective function is used as a stopping criterion.

Originally, the benchmark SOCO 2011 sets the maximum dimension at $D = 1000$. To increase the dimension, instead of shifting the functions as provided by the benchmark, we shifted them randomly between the interval $[\frac{L}{10}, \frac{U}{10}]$, where L is the lower-bound and U is the upper-bound.

The Figure 3.9 shows the obtained SpeedUps. Overall SpeedUps are higher than in the case of the dimension $D = 1000$ (Figure 3.6), except for the function F_1 Shifted Sphere. This can be explained by the nature of this problem (separable without local optima), both algorithms converge quickly to the optimum (it is known here). Therefore ILS

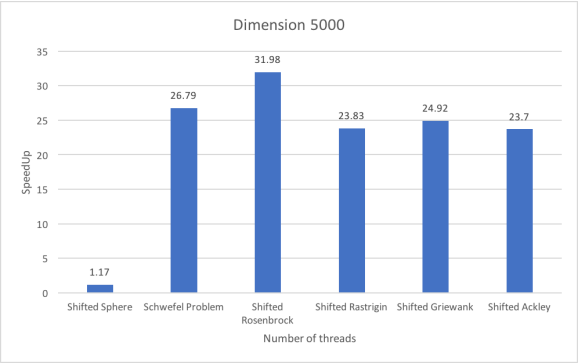


FIGURE 3.9: SpeedUp at dimension D = 5000 for the first six functions of the benchmark SOCO 2011 with number of threads $n = 64$.

is lost in trying to explore intensively, hyperspheres where an optimal or near-optimal solution was already found. It is important to mention that this happens only if the stopping criterion is based on evaluation numbers. If PFDA is configured to stop when a target solution is found, then, the SpeedUp would be significantly higher.

3.2.3 Proposed Multi-Nodes Implementation - MA-FDA

To understand the motivation behind developing a multi-node version of FDA it is important to recall the structure of the approach. At each level FDA evaluates all hyperspheres and select only the best one to be further decomposed. Once the maximum depth, k is reached, ILS is triggered to exploit each hypersphere. However, once all k-th level hyperspheres have been exploited, if the stopping criterion has not been reached, FDA has a backtracking procedure (Section 2.4.2) to move up the tree and explore other parts of the search space. As a recall, Figure 3.1 shows the main life cycle using UML. In both the sequential version and PFDA, diversity is maintained by evaluating all hyperspheres at each level and backtracking in the search tree.

Maintaining diversity in the search space is a challenge that every metaheuristic faces while looking for the global optimum. In our case, while exploring the search space FDA builds a tree of hyperspheres and as mentioned, select only certain branches to be evaluated and further exploited.

While removing the bottlenecks to increase the FDA's speed was the main focus when developing PFDA, improving its diversity was the primary concern when designing the multi-node version called "Multi-Agent Fractal Decomposition Algorithm" or MA-FDA.

Theoretically, if no stopping criterion was set, FDA would explore the entire tree of hyperspheres built. In practice, all algorithms have a stopping criterion weather it is

time or number of function evaluations, leaving in our case a whole part of the tree unexplored.

The main idea behind MA-FDA was to allow each computer node in the cluster to explore a different branch in the tree. In our algorithm each node would select, at a given level, a different hypersphere to be decomposed further.

The initialization phase and evaluation of all the hyperspheres remain the same until a given level l is reached (lesser than the maximum depth k). At this level, after the evaluation and sorting of all hyperspheres, each node would select the hypersphere corresponding to their order number in the cluster. For instance, if $l = 1$, at the first level each node, from 1 to N, would select the sorted corresponding hypersphere. The first node would take the first hypersphere, the second node would select the second hypersphere and so on until the last computer node available select its assigned hypersphere. This is motivated by the fact that hyperspheres are sorted according to their potential as per 2.4.1. It was only logical to let the nodes select them in order of quality. If the number of computer nodes is lesser than the number of hyperspheres, $2 \times D$ in our case, then the remaining will be stored and explored further using the backtracking procedure. Once the stopping criterion is reached on each instance, the master node will collect all best solutions found by each node and give the best one as the final result.

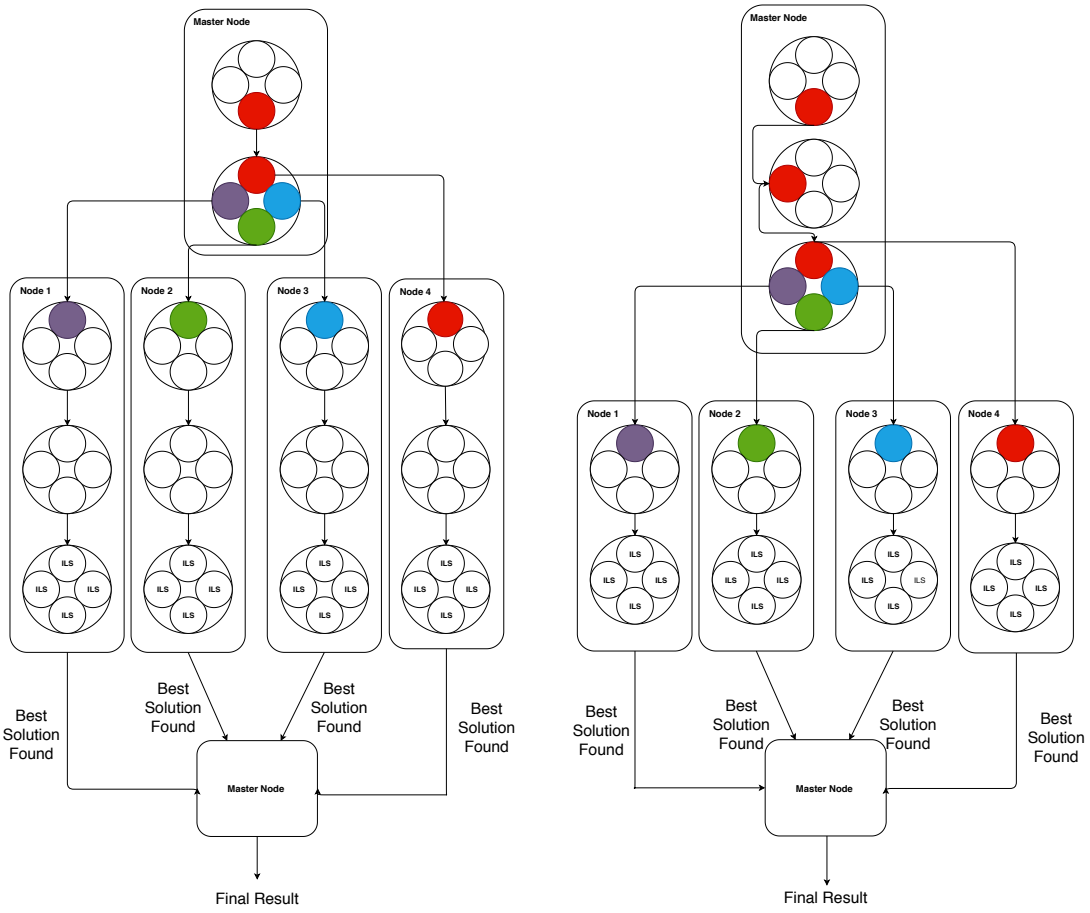
Given N number of computer nodes, MA-FDA will be able to explore N different branches from the given level k. This would have the effect to improve significantly the diversity of FDA from the given level l. The study of the effect of the level choice l is given in the Section 3.2.4 related to experimentations. The main principle of MA-FDA is illustrated in Figure 3.10.

Given this procedure, we have decided to explore two different variations:

- MA-FDA-S1: Each node benefit from the same amount of function evaluations. In our experimentations, we have chosen to compare MA-FDA with its original version where the stopping criterion was set to $5000 \times D$.
- MA-FDA-S2: Each node only benefits from a fraction of the function evaluations, hence $\frac{5000 \times D}{N}$. At maximum depth k, ILS performs only one iteration, meaning it exploits each dimension only once. After this iteration, all nodes report back to the master node and only the best instance is given the remaining function evaluation to pursue the exploitation phase where the others are terminated.

From a technical point of view, the framework OpenMPI[2] was our final choice for the implementation of the multi-node of FDA. Compatible with the original implementation

[2]Open MPI 3.0 is used in this thesis



(a) MA-FDA Parallelized from l = 2 (b) MA-FDA Parallelized from l = 3

FIGURE 3.10: The main life cycle for MA-FDA with N = 4, D = 2 and (a) l = 2 and (b) l = 3.

of our algorithm in C++, it is commonly used in the literature for multi-node environment [Cung et al., 2001] as well as popular distributed frameworks such as ParadisEO [Cahon et al., 2004].

3.2.4 Results and discussions of MA-FDA

To assess the performance of MA-FDA we have compared it with the original results of FDA presented in Section 2.5.5. As a recall, tests were performed on 19 functions (F_1-F_{19}) for large-scale continuous optimization taken from the special issue of Soft Computing on Scalability of Evolutionary Algorithms (SOCO 2011). MA-FDA will be assessed both on the precision and speed.

As mentioned, both variations are evaluated MA-FDA-S1 and MA-FDA-S2. The cluster available was composed of machines with the following characteristics: one 3.1 GHz Intel Xeon® Platinum 817 processor with 16GB of RAM.

3.2.4.1 MA-FDA-S1

MA-FDA-S1 is the first MA-FDA's variation evaluated. In this version, all N nodes have the same number of functions evaluations as stopping criterion *i.e.* $5000 \times D$. In this case the original stopping criterion is not respected and is indeed multiplied by the number of computer nodes used for the experimentation. In our case we have tested MA-FDA-S1 with $N = 2$, $N = 5$, $N = 25$ *and* $N = 50$ for dimension $D = 50$.

As the stopping criterion has changed we have run the original FDA with an extended number of functions evaluation as stopping criterion equal to $N \times 5000 \times D$. The experimental results obtained by MA-FDA-S1 are shown in Table B.3 for $N = 2$ in Table B.4 for $N = 5$, in Table B.5 for $N = 25$ and in Table B.6 for $N = 50$. To compare the different approaches we have used the Friedman Rank Sum score. The ranks are shown in Table B.7.

MA-FDA integrates a new parameter which is the level at which the computer nodes takes their independence and explore a different branch of the search tree. It is important to study the sensitivity with respect to the parameter l. Among the different levels shown in Tables from B.3 to B.6, the level $l = 3$ is undoubtedly the choice for the parameter. Indeed, MA-FDA-S1 with level $l = 3$ finds the best solution for the function *F3 - Shifted Rosenbrock*, one of the main challenge for FDA. In addition, the rank of the level $l = 3$ is confirmed using the Friedman Rank sum score in Table B.7.

From a precision point of view, it is clear that the original version FDA shows the worst performance as its number of function evaluation is limited compared to MA-FDA-S1. To compare with the same stopping criterion, the original version with an extended number of function evaluation has been tested and shows that it outperforms MA-FDA-S1. This shows that the original version navigates well into the search tree to find the best solution possible. However, the time needed by FDA with an extended number of function evaluation grows linearly with N where the computing time for MA-FDA-S1 as N grows remains stable as shown on Table B.2 and on Figure 3.11 (times have been averaged over 20 independent runs). It is interesting to highlight the fact that on one of the composed functions *F9 - CompF9 F3 025*, MA-FDA-S1 obtains better results than FDA.

3.2.4.2 MA-FDA-S2

As mentioned, MA-FDA-S2 differs from the first version in that all nodes share a "common pot" of function evaluations. The motivation behind this version was to compare

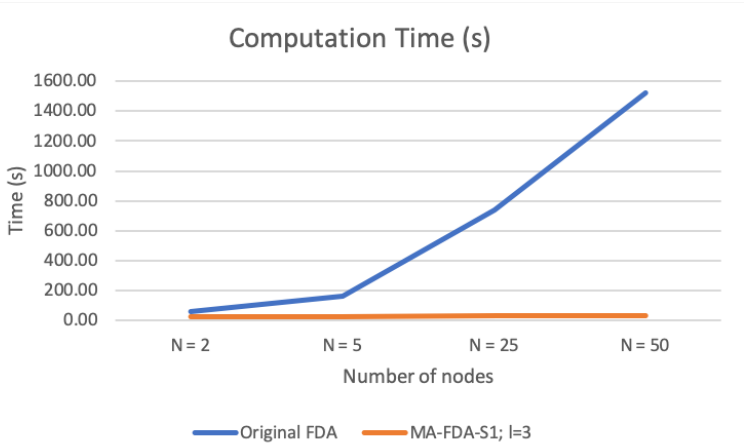


FIGURE 3.11: Computation time required to solve the 19 functions of the benchmark used. Times are in seconds.

MA-FDA with its original version using the same stopping criterion *i.e.* $5000 \times D$ function evaluations. Each node has then $\frac{5000 \times D}{N}$ function evaluations.

Similar to MA-FDA-S1, computer nodes select a different hypersphere at the given level l and pursue the exploration of their respective branches. Once the maximum depth k is reached, ILS is triggered in the first hypersphere and will perform one iteration on each node N and go over each dimension only once. All nodes will report back to the master node their results after this iteration and only the one with the best results will be allowed to continue and consume all remaining functions evaluations to reach the global stopping criterion.

The approach's sensitivity to its parameter level l have been tested with $N = 2$, $N = 5$, $N = 25$ on $D = 50$. Experimentals results are given in Table B.8 for $N = 2$ in Table B.9 for $N = 5$ and in Table B.10 for $N = 25$. To compare the different approaches we have again used the Friedman Rank Sum score. The ranks are shown in table B.11. As per the Rank Sum score, the sensitivity of MA-FDA-S2 with regards to the parameter l is the same as MA-FDA-S1 and Table B.11 highlights that $l = 3$ is the best choice for this parameter.

From the experimental results, MA-FDA-S2 outperforms FDA when the number of computer nodes is small. Indeed, as N increases, the number of functions evaluations consumed by different nodes grows as well, hence it limits the exploitation of the hyperspheres by the best node after the first ILS iteration. Time-wise, MA-FDA-S2 has a similar execution time as FDA. The only time overhead comes from synchronising the different nodes which can be neglected here as the number of communication is limited. The original version of FDA takes (averaged over 20 runs) 30 seconds to solve the 19

functions on dimension $D = 50$ and MA-FDA-S2 takes also 30 seconds (averaged over 20 runs).

3.3 Conclusion

In this work, two different approaches have been studied. First, a parallel version of the FDA algorithm, called PFDA, was proposed. The algorithm has been extensively tested on the SOCO 2011 Benchmark on large scale problems, going from 50 to 5000. Based on the SpeedUp criterion, it is easy to see that parallelizing FDA has improved significantly its performances on large-scale problems. However, during the exploration phase, PFDA consumes a lot of functions evaluations. When the stopping criterion is a target value of the objective function a high SpeedUp was obtained. It can be concluded that this new approach enforces the original strengths as it converges significantly faster. However, PFDA remains less efficient in the case of highly non-separable problems. This is due to the heuristic used to explore hyperspheres, ILS.

Following PFDA, our approach of FDA was modified to run on distributed IT infrastructure. This new approach called "Multi-Agent Fractal Decomposition Algorithm" (MA-FDA) was introduced. Two different variants were studied. Both were compared to the original version, FDA, on the same functions 19 functions taken from the SOCO Benchmark. The first version, MA-FDA-S1 benefits from an extended number of function evaluations. Performance-wise, with an increased stopping criterion, the original version has better performance. However, the time required to solve grows linearly with N (the number of nodes). This is when MA-FDA-S1 shines, having a stable computing time regardless of the number of nodes and therefore performing better if time is set as a stopping criterion. The second version, MA-FDA-S2 is designed with function evaluations as the main concern. It's running time is equivalent to the original version but offers a better diversity and some improvement can be found when the number of nodes is not too large. From experimentations, it can be concluded that MA-FDA benefits from multi-node environments regardless of the size and improves significantly the diversity of the original version FDA.

Chapter 4

Design of Fractal Decomposition based algorithm for multi-objective optimization

4.1 Introduction

In Multi-objective Optimization Problems (MOP) the goal is to optimize at least two objective functions simultaneously. In this chapter, we are interested in using FDA to deal with MOPs because in the literature decomposition-based algorithms have been successfully applied to solve these problems.

We propose to extend FDA to solve MOP problems using two different approaches:

- Mo-FDA-S: Scalarization approach
- Mo-FDA-D: Dominance and Indication

Mo-FDA-S adapts FDA using a scalarization technique. This approach has also been developed to benefit from a multi-node environment to improve the computational time taken to solve MOPs problems. This chosen architecture benefits from containers, lightweight virtual machines that are designed to run a specific task only.

The second approach, Mo-FDA-D uses the principle of non-dominated sorting to find the best Pareto Front possible. Mo-FDA-D has changed at its core the principle of FDA and proposes both a new hypersphere evaluation technique based on the evaluation of the hypervolume and a new local search algorithm ILS.

The chapter is organized as follows. The next Section 4.2 presents the first approach Mo-FDA-S, the chosen scalarization method and multi-node architecture. Section 4.3 presents the second algorithm Mo-FDA-D. In Section 4.4 the experimental settings and results against competing methods are detailed. Finally, Section 4.4 concludes this chapter.

4.2 Mo-FDA Scalarization: Mo-FDA-S

As mentioned, in a multi-objective problem, multiple objective functions are being optimized at the same time. However, FDA has originally been designed to solve mono-objective problems. Thankfully, in the MOP literature, scalarization methods can be found to transform a multi-objective problem into a single objective problem. The main idea behind scalarization is to associate each objective function with a weighting coefficient and minimize the sum of all weighted objective functions. Several methods can be found [López Jaimes & Zapotecas-Martínez, 2011] and we introduce two of the most popular approaches that we used in our experimental studies.

4.2.1 Weighted Sum

This approach consists of using a weight vector $\omega = (\omega_1, ..., \omega_k)$, to combine the k objective functions, solving as follows:

$$\begin{aligned} &minimize \sum_{i=1}^{k} \omega_i f_i(x) \\ &subject\ to\ x \in X \end{aligned} \tag{4.1}$$

with $\omega_i \geq 0\ for\ i = 1, ..., k$ and $\sum_{i=1}^{k} \omega_i = 1$. The set of non-dominated solutions can be generated by using different weight vectors ω in using the weighted sum approach. In the case where the Pareto Front is convex (or concave in case of maximization), this technique works well [Zhang & Li, 2007]. However, it is not always the case when optimizing multi-objective problems. This is why we have chosen to study another scalarization technique, know as the Tcheybycheff approach [Miettinen et al., 2008] which allows overcoming this issue.

4.2.2 Tcheybycheff Approach

This technique as the particularity to introduce the notion of ideal point or reference point z_i^* as follows:

$$\begin{array}{ll} Minimize & \max\limits_{i=1,...,k} \quad [\omega_i(f_i(x) - z_i^*)] \\ \text{Subject to} & x \in X \end{array} \tag{4.2}$$

with k the number of objective functions to optimise, $z^* = (z_1^*, ..., z_k^*)$ the reference point with z_i^* the optimum of function f_i and as in the previous method, $\omega_i \geq 0$ for $i = 1, ..., k$ and $\sum_{i=1}^{k} \omega_i = 1$. One weakness in this technique is that the aggregation obtained with the vector ω is not smooth for a continuous problem [Zhang & Li, 2007]. However, this is not an issue as our algorithm does not need to compute the derivate of the aggregation function.

4.2.3 Proposed Approach and Parallelized Architecture

The first approach proposed to solve multi-objective problems is called "Multi-Objective Fractal Decomposition Algorithm Scalarization" or Mo-FDA-S. It uses the Tcheybycheff function to transform a MOP into a mono-objective problem. By using N different weight vectors ω, Mo-FDA-S solves N different problems, each generating one point composing the final Pareto Front (PF). The algorithm used to solve each problem is the same as presented in Chapter 2 and described in Algorithm 5.

One of the downsides of using scalarization methods is that the number of points composing the PF found by the algorithm will be, at most, the same as the number of different weight vectors N. In certain cases, if two or more weight vectors are too close, the algorithm might find the same local optimum for the different weight vectors ω. This is in opposition to other MOP algorithms which are based on other techniques such as non-dominated sorting, for instance, NSGA-II [Deb et al., 2002b]. One of the consequences is that, if the stopping criterion is based on a maximum of function evaluations Max_{fe}, each instance of the algorithm will only benefit from $\frac{Max_{fe}}{N}$ function evaluations. This will have the effect of limiting the potential for each instance to find the global optimum for each weight vector ω.

4.2.3.1 Proposed architecture

As mentioned, scalarization has been used to adapt the original version to solve multi-objective problems. Using this technique, to obtain N points in the Pareto Front, the

algorithm will be launched N times with N variation of the weight vector ω as showed in Equation 4.2. If we consider that the number of Function Evaluations (FE) is used as a stopping criterion, even though, each instance only has $\frac{Max_{fe}}{N}$ FE, the computational time can increase significantly. To overcome this, a multi-node architecture has been developed for Mo-FDA-S. The idea is to have each node finding one point corresponding to one combination of the weights ω and combine all their results to build the full final Pareto Front. This idea is inspired by the work presented in MA-FDA-S1 in Section 3.2.4.1. The challenge behind this architecture is that the computing resources needed increase with the size of the Pareto Front. For instance, if one were to set $N = 100$ points, it means that 100 nodes would be required, hence 100 different computers (or virtual machines), which can be seen as an oversized architecture. To tackle this important issue we have decided to develop the approach using *containers* and specifically the powerful combination of *docker* as the container technology with *kubernetes* as the orchestrater as shown on Figure 4.2. Containers are significantly lighter than virtual machines as they all share the same operating system kernel. This way, a single machine can host a lot more containers than virtual machines. This architecture is significantly lighter than a traditional one and allows to benefit from multi-node approaches while developing it on a limited number of hosts. In addition, containers can be deployed on multiple different physical (or virtual) machines seamlessly, without having to change the structure of our algorithm. Kubernetes is the leading open-source solution for container-orchestration and takes care, in our case, of the creation and deployment of all the containers on the different hosts without changing anything in the algorithm implementation.

An example of computation time is shown in Table C.1 and on Figure 4.1. This example considers the time to solve a function in dimension D = 30 with 100 different points (*i.e.* 100 different weights vectors ω) on N different virtual hosts with $N = 2$; $N = 10$; $N = 25$ *and* $N = 50$. It is important to indicate that even on two hosts (N = 2), a computation gain can be observed, however not as important as when the number of hosts increases. This is because each host has to handle, in this case, 50 different containers. Moreover, when N increases significantly, here N = 25 and N = 50 the gain in time is significant compared to the sequential version but at some point, the increase in compute nodes does not decrease the computational time. This is due to the communication overhead required to synchronise all nodes and gather all points compositing the PF. All tests have been done on a cluster with machines with the following characteristics: one 3.1 GHz Intel Xeon® Platinum 817 processor with 16GB of RAM. Moreover, Mo-FDA-S has been developed in Python and similar to MA-FDA-S1 uses the library MPI for the multi-node implementation.

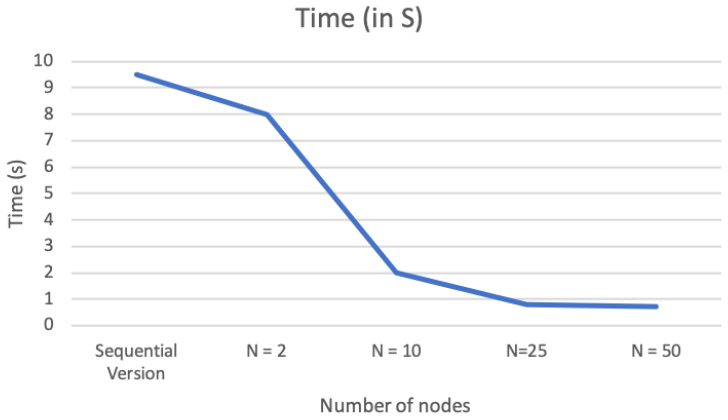


FIGURE 4.1: An example of computation time required to solve a function with Mo-FDA-S with the different number of physical nodes N for 100 instances of FDA, hence 100 points in the Pareto Front. Times are in seconds.

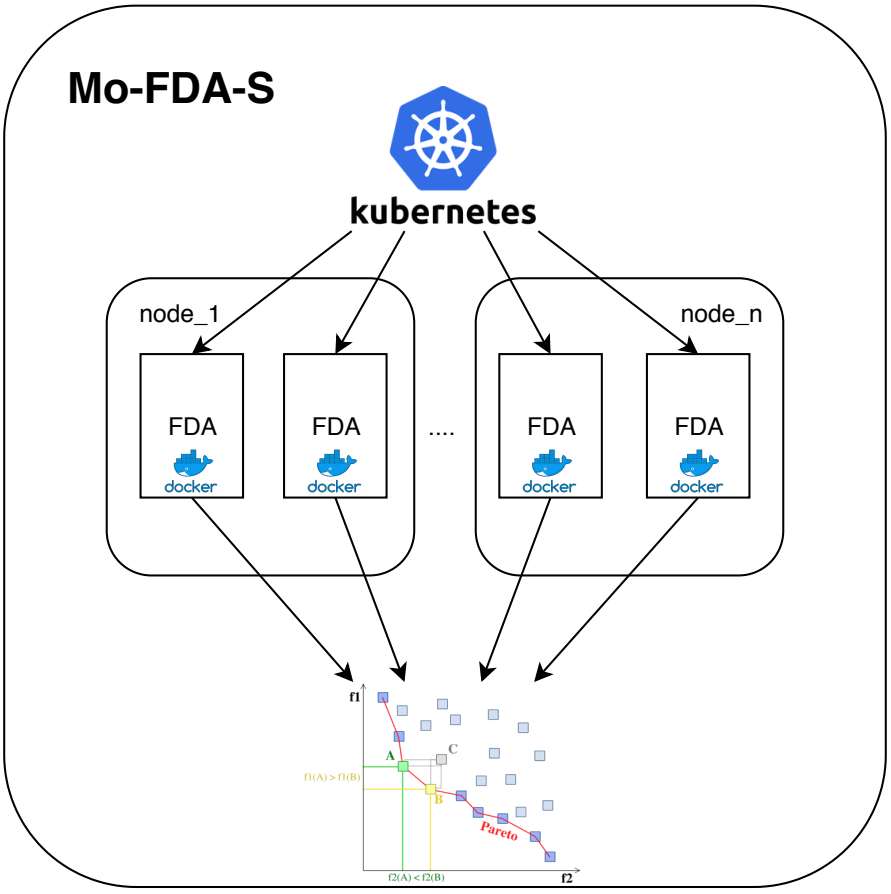


FIGURE 4.2: The architecture of Mo-FDA-S using containers on N different nodes.

4.3 Mo-FDA Dominance

In line with our objective to adapt FDA to solve multi-objective problems, Mo-FDA-S has been developed using scalarization methods. In this section, we present another approach we have developed to solve MOP problems but inspired by non-dominated sorting approaches such as the well known NSGA-II [Deb et al., 2002b]. This new approach is called "Multi-Objective Fractal Decomposition Algorithm Dominance Based" or Mo-FDA-D.

The idea behind Mo-FDA-D is to keep the structure of the framework provided by the original version [Nakib et al., 2017], *i.e.* the geometric fractal decomposition (detailed in Section 2.2) as well as the geometric form and the different phases composing FDA. The initialisation phase (described in Section 2.2) remains the same. The order of the other main phases is also respected, *i.e.* the hypersphere evaluation (Section 2.4.1), local search *ILS* (Section 2.4.3) and the backtracking (Section 2.4.2). However, the procedure to evaluate hyperspheres and the heuristic to conduct the local search at the fractal depth have been adapted to multi-objective problems.

4.3.1 Multi-objective Promising hypersphere selection (Exploration strategy)

As per the original approach, this procedure aims to select the most promising region that might contains the best solution. In the context of multi-objective problems, a single solution does not exist and a set of solutions composing the Pareto Front (PF) corresponding to the set of Pareto optimal solutions is the final solution of a MOP problem.

In this new version, the aim is to find both the most promising region to be further decomposed but also to find potential non-dominated points composing the final Pareto Front (PF). To do so, we evaluated multiple points along each dimension as per the following equations:

$$\vec{s} = \vec{C}_l \pm \frac{r_l}{\gamma} \times \vec{e}_i \quad \text{for} \quad i = 1, 2, ..., D \tag{4.3}$$

where $\vec{C}^{(l)}$ is the coordinates of the center of hypersphere being evaluated, r is its radius, $\gamma \in [1, 3]$ and $\vec{e}_i$ is the unit vector at the dimension i.

This is illustrated on Figure 4.3 with a two-dimensional example in two different scenarii:

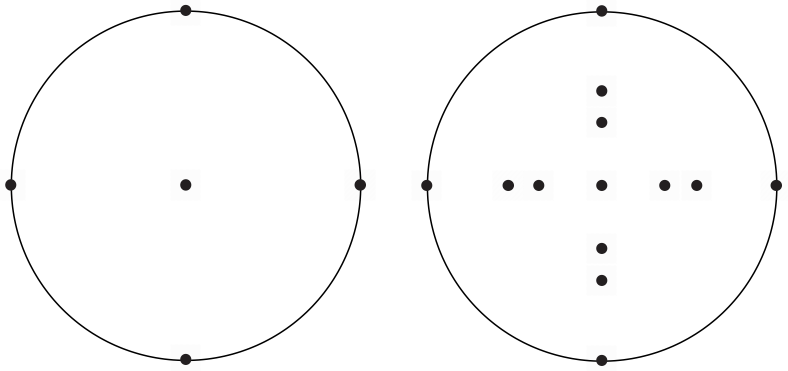


FIGURE 4.3: Evaluating (a) $2 \times D$ points plus the center with $\gamma = 1$ and (b) $6 \times D$ points plus the center with $\gamma = \{1, 2, 3\}$.

- (a) $2 \times D$ points plus the center are evaluated with $\gamma = 1$, meaning that the points are on the sphere.
- (b) $6 \times D$ points plus the center are evaluated with $\gamma = \{1, 2, 3\}$. Points are on and within the hypersphere.

All evaluated points are stored in a temporary list to be sorted. The sorting is based on the Pareto Dominance and only non-dominated points are kept, producing a local Pareto Front of locally non-dominated solutions within the hypersphere. The sorting algorithm used to sort evaluated points and generate the local PF is called *Simple Cull* and is described in [Geilen & Basten, 2007].

Once the local PF obtained, all points are compared to the *nadir point* of the objective space and all points above are excluded from the PF. As a recall the *nadir point* is defined as $z_i^{nad} = max\{f_i(x)|x \in PF\}$. In other words, z^{nad} defines the upper bound of the Pareto Front. This is why all the points above are discarded.

Where in the original FDA, the quality score of the hypersphere is defined as the maximum slope as defined in Equation 2.34, in this case, we had to define a new quality score. We have decided to chose the hypervolume as quality metrics. To do so, we compute the hypervolume of the local PF with regards to the general *nadir point*, z^{nad}. Details of the algorithm used to compute the hypervolume can be found in [Nowak et al., 2014].

As mentioned, the higher the hypervolume the better the Pareto Front is (in case of minimization), therefore the hypersphere with the highest value is considered better than the other hyperspheres of the same level and will, therefore, be selected to be further decomposed. The intuition behind the use of the hypervolume is that some of

the locally non-dominated solutions found within the hypersphere are more likely to be part of the final PF.

Once all the hyperspheres of a given level have been evaluated, all the locally non-dominated sets are concatenated and sorted again to find once global PF of non-dominated solutions.

4.3.2 Multi-objective Intensive Local Search (ILS)

Once the fractal depth k is reached, the local search is triggered. In this context, it is important to notice that different local searches or metaheuristic could be used at this step. We have chosen to adapt our existing ILS to MOP.

In this adapted version instead of searching locally within hyperspheres of the last level, ILS iterates around each non-dominated solutions found so far during the exploration phase of evaluating hyperspheres. Therefore, the entry point of one ILS instance is one point of the current global Pareto Set.

ILS starts by creating two empty lists, one for the Pareto Set $listNewPS$ (decision space) and one for their solutions in the Pareto Front denoted $listNewPD$ (objective space) and place in the first one the point given as input parameter.

Then for each dimension and each point in $listNewPS$, ILS will produce two additional points denoted $\vec{x}_L$ and $\vec{x}_R$. They stand in opposite directions from the current point being exploited, $\vec{x}_C$ at equal distance ω, also called step size as per the following equations:

$$\vec{x}_L = \vec{x}_C + \omega \times \vec{e}_i \tag{4.4}$$

$$\vec{x}_R = \vec{x}_C - \omega \times \vec{e}_i \tag{4.5}$$

where $\vec{e}_i$ is the unit vector which the i^{th} element is set to 1 and the other elements to 0,

$\vec{x}_C$, $\vec{x}_L$ and $\vec{x}_R$ are then evaluated and placed in $listNewPS$ and their solutions in the objective space, respectively $F(\vec{x}_C)$, $F(\vec{x}_L)$ and $F(\vec{x}_R)$ are added to the list of solutions ($listNewPF$).

Once all the points in $listNewPS$ have been exploited for the current dimension, the same sorting algorithm used in Section 4.3.1 is applied to the list of all potential solutions ($listNewPF$) and this will generate a new local Pareto Front of non-dominated solutions. Therefore, $listNewPF$ now only contains a set of non-dominated solutions and $listNewPS$ only contains their equivalent in the search space.

At the end of each iteration, once all dimensions have been searched, ω is multiplied by a coefficient defined as a hyperparameter of Mo-FDA-D.

This is repeated until either:

- The stopping criterion is reached and Mo-FDA-D is done;
- or ω has reached its minimum value ω_{min} and therefore ILS moves on to the next point in the Pareto Set found during exploration.

Once ILS has finished searching around each point of the PS, all points found during ILS research are sorted and only the non-dominated points will remain and will compose the new global Pareto Set. In this case, either the stopping criterion is reached and Mo-FDA-D has finished or the backtracking procedure describe in Section 2.4.2, is applied and a new sphere from the level *k - 1* is selected to be decomposed. The whole procedure is illustrated in Algorithm 9.

4.4 Results and Discussions

In this section, the two proposed algorithms to solve MOP, Mo-FDA-S and Mo-FDA-D are analyzed and their performance is exposed using different functions taken from well know benchmarks. First, only the different approaches are compared using a simple function followed by a comparison with competing algorithms found in the literature is conducted.

4.4.1 Benchmark Functions

The first function used to compare our two approaches was the well known "Fonseca–Fleming" problem [Fonseca & Fleming, 1995]. Its Pareto Front is shown on Figure 4.4 and the function is defined in Equation 4.6.

$$\text{FON}: \begin{cases} \text{Minimize} & f_1(X) = 1 - exp\left[-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right] \\ \text{Minimize} & f_2(X) = 1 - exp\left[-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right] \\ \text{Domain} & -4 \leq x_i \leq 4 \quad , \quad i = 1, 2, \cdots, n \end{cases} \tag{4.6}$$

We then used the ZDT [Zitzler et al., 2000] and DTLZ [Deb et al., 2002a] functions to compare our approaches with competing algorithms.

Algorithm 9: ILS procedure

Input: $\omega_{\min} = 10^{-5}$. //precision or tolerance error
Input: Coefficient step-size: λ
Input: D // the dimension of the problem
Input: Number of function evaluations $NBEval$
Input: The first point to search as starting point $startingPoint$
Set the step size ω to the radius of a $k_t h$ level hypersphere H
Set an empty list for Non-dominated Points Coordinates $listNewPS$
Add $startingPoint$ to $listNewPS$
Set an empty for Non-dominated Points Solutions $listNewPF$
while $\omega \geq \omega_{min}$ **do**
  for *Each dimension* $i = 1, \ldots, D$ **do**
    foreach $currentPoint \in listNewPS$ **do**
      set $\vec{x}_C$ = currentPoint
      $\vec{x}_L = \vec{x}_C - \omega \times \vec{e}_i$
      $\vec{x}_R = \vec{x}_C + \omega \times \vec{e}_i$
      Evaluate the fitness of $\vec{x}_C$, $\vec{x}_L$ and $\vec{x}_R$
      NBEval = NBEval + 3
      Add $F(\vec{x}_C)$, $F(\vec{x}_L)$ and $F(\vec{x}_R)$ to $listNewPF$
    end
    Sort $listNewPF$ to leave only the non-dominated solutions
    Modify $listNewPS$ so it contains only the coordinates of the non-dominated solutions
  end
  Decrease the step size ω: $\omega = \omega \times \lambda$.
end
Output: $listNewPS$ and $listNewPF$

4.4.2 Sensitivity analysis of the multi-objective algorithms

In this subsection, we aim to analyse the sensitivity of our algorithms against its parameters.

4.4.2.1 Parameters sensitivity of Mo-FDA-S

As Mo-FDA-S is based on the original version, the analysis has been done in Section 2.5.3. However, as mentioned in Section 4.2, two scalarization methods can be used, the Weighted Sum or the Tcheybycheff. We have conducted a comparison to test both techniques. To conduct this analysis, we have chosen to use the "Fonseca–Fleming" problem [Fonseca & Fleming, 1995] as per Equation 4.6 for dimensions $D = 2; D =$

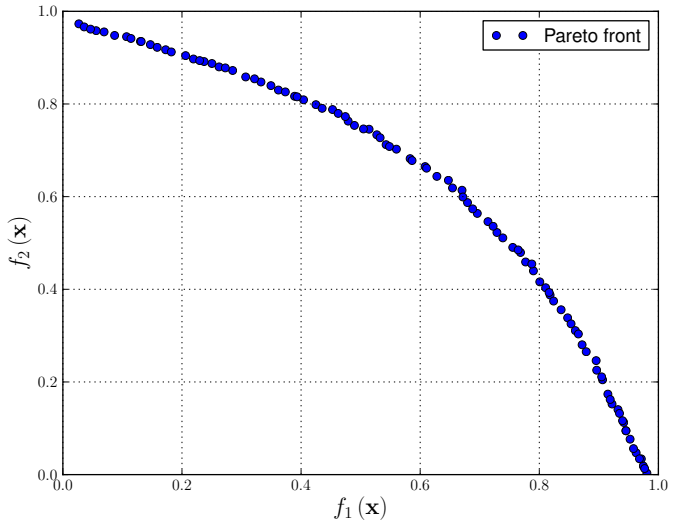


FIGURE 4.4: Pareto Front for the Fonseca–Fleming problem.

5; $D = 10$ and $D = 30$. The stopping criterion has been set up to $5000 \times D$ and 10^{-5} as precision tolerance ω_{min}. The hypervolume has been chosen to be the metric to compare performances (Figure 1.5).

From the results shown in Figure 4.8 and the Hypervolumes in Table C.4, it is obvious that the Tcheybycheff method is the most performant one. Besides, it is interesting to note that Mo-FDA-S works well at low dimensions and high dimensions but is less performant on intermediate dimensions.

4.4.2.2 Parameters sensitivity of Mo-FDA-D

As Mo-FDA-D has been modified at its core, this subsection aims to analyze with regards to three parameters:

- The number of points evaluated in the hypespheres as defined in Section 4.3.1.
- The fractal depth k.
- The step size λ by which ω is multiplied in ILS as detailed in Section 4.3.2.

Besides, as per our research, we have also analysed the need for the local search, ILS, in Mo-FDA-D. To conduct this analysis, we have chosen to use the "Fonseca–Fleming" problem [Fonseca & Fleming, 1995] as per Equation 4.6 and the hypervolume as metrics to compare the performances (Figure 1.5)

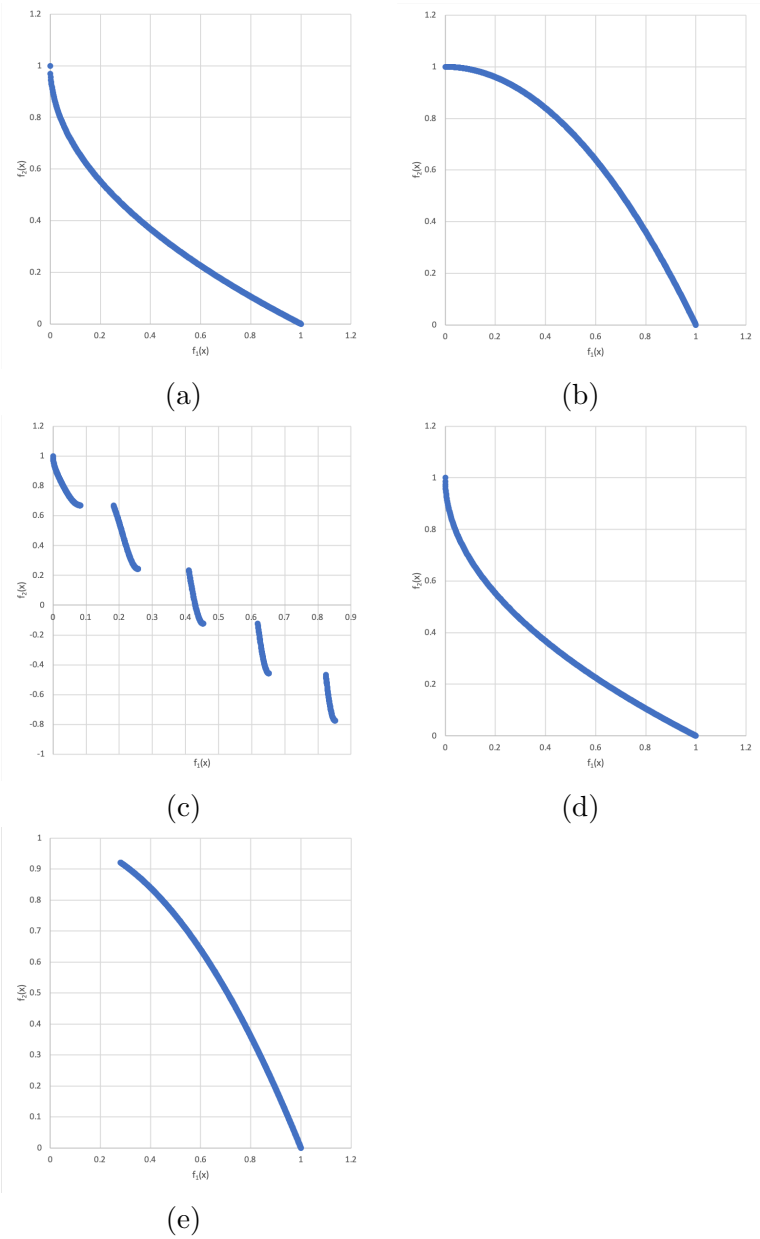


FIGURE 4.5: The true fronts for the ZDT functions used in our study. (a) ZDT1, (b) ZDT2, (c) ZDT3, (d) ZDT4, (e) ZDT6.

It is important to highlight the fact that in the different cases, the common criteria were the number of function evaluations set up to $5000 \times D$ as stopping criterion and 10^{-5} as precision tolerance ω_{min}.

We have started the following scenarii for dimension $D = 2$:

- Case 1: 2 points evaluation for hypersphere ($\gamma = 1$); k = 5; Without ILS.
- Case 2: 2 points evaluation for hypersphere ($\gamma = 1$); k = 5; With ILS.
- Case 3: 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$); k = 5; Without ILS.

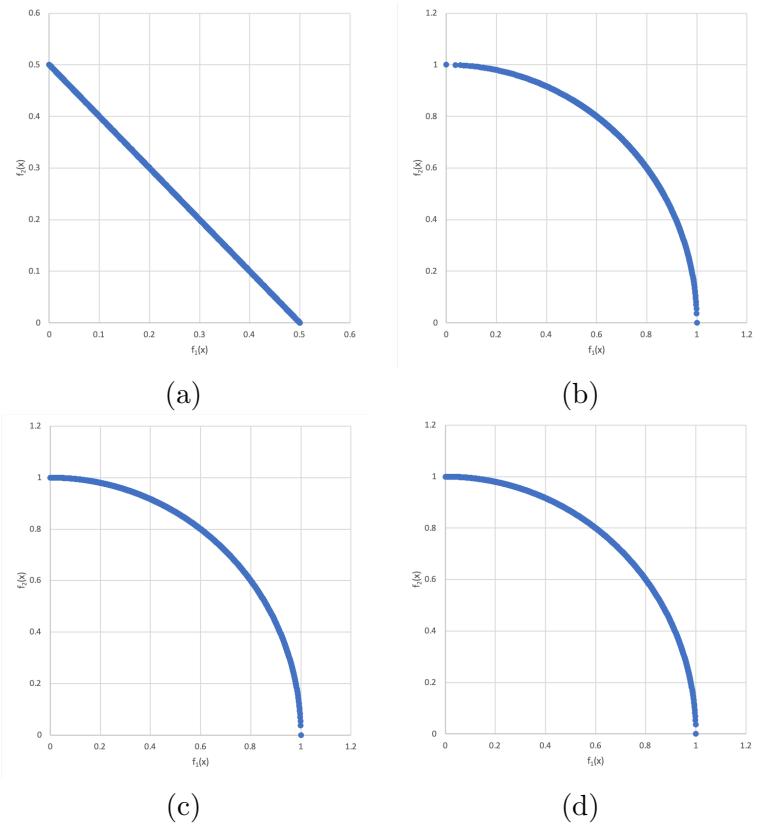


FIGURE 4.6: The true fronts for the DTLZ functions with 2 Objectives used in our study. (a) DTLZ1, (b) DTLZ2, (c) DTLZ3, (d) DTLZ4.

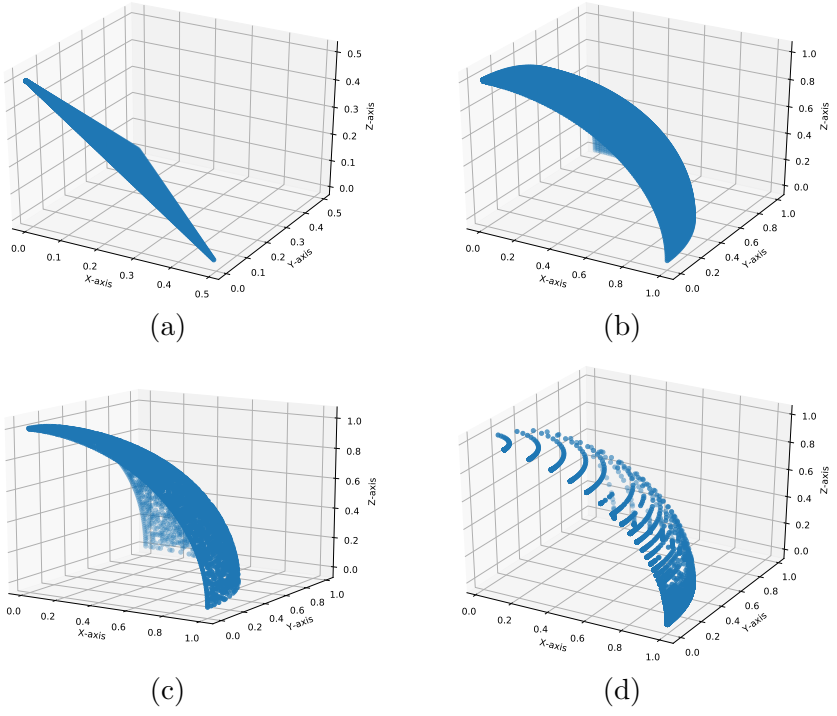


FIGURE 4.7: The true fronts for the DTLZ functions with 3 Objectives used in our study. (a) DTLZ1, (b) DTLZ2, (c) DTLZ3, (d) DTLZ4.

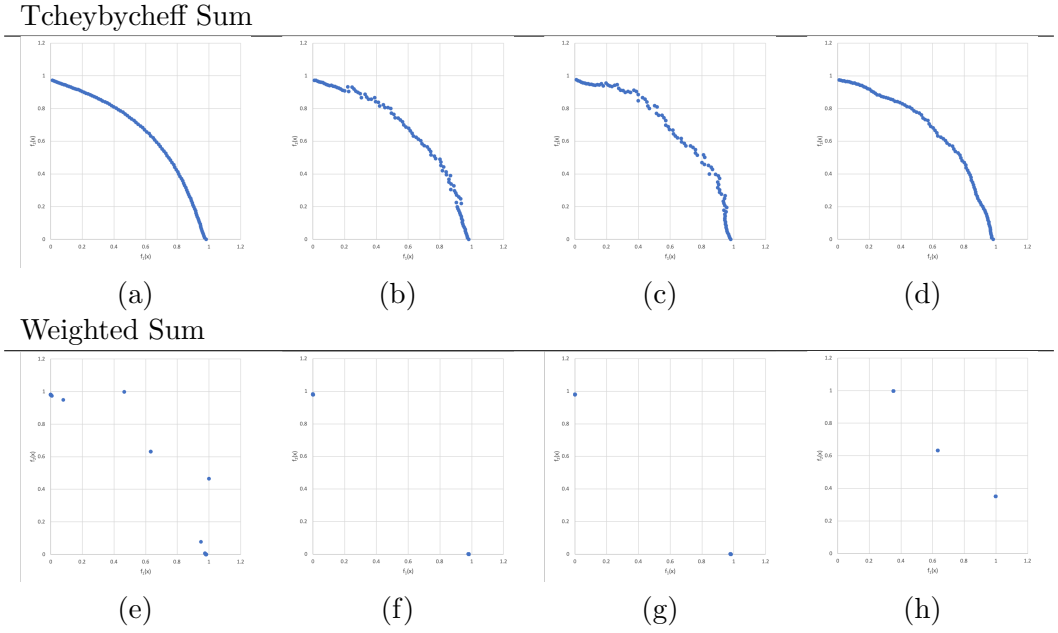


FIGURE 4.8: Pareto Fronts for the two studied scalarization methods on 4 different dimensions. Using Tcheybycheff Sum (a) D=2, (b) D=5, (c) D=10, (d) D=30. Using Weighted Sum (e) D=2, (f) D=5, (g) D=10, (h) D=30.

- Case 4: 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$); k = 5; With ILS.
- Case 5: 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$); k = 8; Without ILS.
- Case 6: 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$); k = 8; With ILS.
- Case 7: 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$); k = 16; Without ILS.
- Case 8: 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$); k = 16; With ILS.

All those scenarii are illustrated on the Figure 4.9 and Results are show in Table C.5. ILS tends to concentrate the Pareto Front around one area and therefore penalise the diversification of the Pareto Front. Going too deep *i.e.* k = 16 decreases the performances of the algorithm, whether ILS is used or not. However, not using ILS increases significantly the computing time. Tuning the parameters impact both hypervolume and computation time. Parameters maximizing the hypervolume lead to an increased running time and vice versa. However, in our study were only interested in the hypervolume. Consequently, from that set of scenarii we can conclude that Case 5 is the best set of parameters.

Those results seem promising but have only been made for the dimension $D = 2$. The chosen benchmark (ZDT and DTLZ) are scaled for dimension $D = 30$. Therefore we have decided to set the parameters as set in Case 5 but with higher dimensions. The following scenarii have been studied:

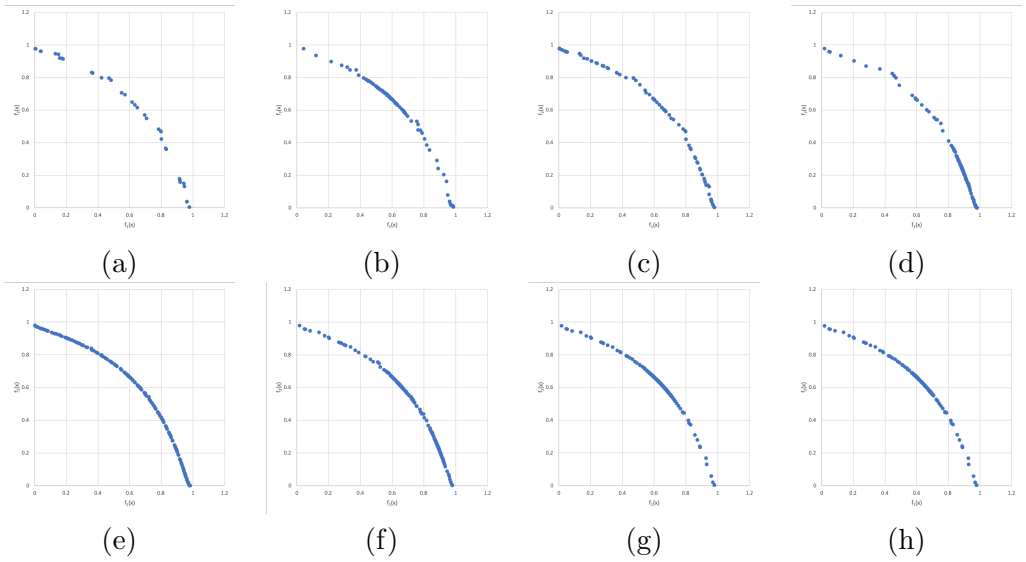


FIGURE 4.9: Pareto Front for the studied Case 1 to 8. (a) Case 1, (b) Case 2, (c) Case 3, (d) Case 4, (e) Case 5, (f) Case 6, (g) Case 7, (h) Case 8.

- Case 9: $D = 5$; Without ILS
- Case 10: $D = 5$; With ILS
- Case 11: $D = 10$; Without ILS
- Case 12: $D = 10$; With ILS
- Case 13: $D = 30$; Without ILS
- Case 14: $D = 30$; With ILS

As mentioned, the other parameters have been set similarly to the Case 5, *i.e.* 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$) and k = 8. The Pareto Sets of the different cases are shown in Figure 4.10 and the quantitative results are shown in Table C.6. Those results highlight the important fact that even though not using ILS works well on low dimensions, it becomes essential to use it as the dimension increases.

4.4.3 Parameter Settings

Following the observations made in the previous section while studying the sensitivity, the approaches with regards to their parameters, we have found empirically that the following parameters work best for the chosen benchmarks, ZDT and DTLZ. For all functions and all solutions, the stopping criterion has been set to 3×10^5 Function evaluations.

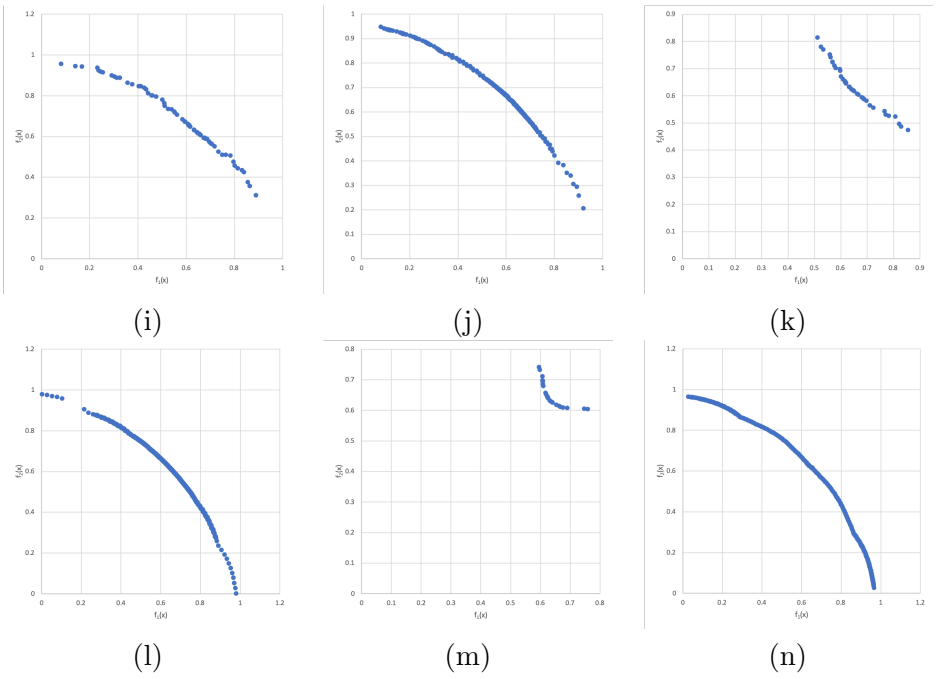


FIGURE 4.10: Pareto Front for the studied Case 9 to 14. (i) Case 9, (j) Case 10, (k) Case 11, (l) Case 12, (m) Case 13, (n) Case 14.

4.4.3.1 Settings for Mo-FDA-S

. For Mo-FDA-S the following parameters were chosen empirically:

For the ZDT benchmark:

- Tcheybycheff as scalarization method
- The fractal deph k = 2
- Coefficient step-size $\lambda = 0.89$
- $\omega_{min} = 10^{-5}$
- Even though the search domain is defined as $x \in [0, 1]$ we have chosen to relax the domain space to $x \in [-2, 2]$ and penalise solutions that are not in the original space. This is due to the fact that as the first hypersphere is initialised in the center of the search space, the initial solution would have been set to $x_i^{init} = 0.5$ for $i = 1, ..., D$. The number of function evaluations per instance would be too small to converge to potential solutions. By relaxing the search space and center it around 0, the initial solution is set to $x_i^{init} = 0$ for $i = 1, ..., D$ and therefore each instance of Mo-FDA-S converges faster.

For the DTLZ 1-3 functions:

- Tcheybycheff as scalarization method.
- The fractal deph k = 2.
- Coefficient step-size $\lambda = 0.5$.
- $\omega_{min} = 10^{-5}$.

For DTLZ 4 function:

- Tcheybycheff as scalarization method.
- The fractal deph k = 3.
- Coefficient step-size $\lambda = 0.2$.
- $\omega_{min} = 10^{-5}$.

4.4.3.2 Settings for Mo-FDA-D

For Mo-FDA-D, the following parameters were chosen empirically:

For the ZDT benchmark:

- The fractal deph k = 5.
- Coefficient step-size $\lambda = 0.5$.
- $\omega_{min} = 10^{-5}$.
- 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$).
- With ILS.

For the DTLZ benchmark:

- The fractal deph k = 5.
- Coefficient step-size $\lambda = 0.2$.
- $\omega_{min} = 10^{-5}$.
- 6 points evaluation for hypersphere ($\gamma = \{1, 2, 3\}$).
- With ILS.

4.4.4 Comparison with competing algorithms

In this subsection, a comparison of our proposed algorithms is conducted with other well known MOP algorithms from the literature. We have considered the following algorithms:

- NSGA-II [Deb et al., 2002b] is a computationally fast and elitist Multi-Objective Evolutionary Algorithms (MOEA) based on a non-dominated sorting approach.
- NSGA-III [Deb & Jain, 2014] is an extension of NSGA-II adapted to solve many-objective problems, *i.e.* more than 3 objectives. It works with a set of supplied or predefined reference points aiming to maintain the diversity among population members.
- MOEA/D-DE [Li & Zhang, 2009]. This algorithm is also an MOEA but based on decomposition. Similarly to Mo-FDA-S, it uses scalarization to transform the MOP into a single-objective problem. The different scalar optimization subproblems are here optimized simultaneously. The Tcheybycheff method is used for 2-objective functions.
- GWASFGA [Saborido Infantes et al., 2017] stands for "Global Weighting Achievement Scalarizing Function Genetic Algorithm". This algorithm is also based on a scalarization method and uses an achievement scalarizing function which is based on the Tcheybycheff method but includes the use of the utopian and the nadir points. GWASFGA generates the weight vectors so that they define an evenly distributed set of projection in the objective space.
- CDG [Cai et al., 2018], also a decomposition-based MOEA. Instead of using a traditional scalarization method such as Tcheybycheff, CDG-MOEA uses a constrained decomposition with grids. One objective function is selected to be optimized while the other objective functions are converted into constraints by setting up the upper and lower bounds.

All experimentations on the competing algorithms have been done using the framework jMetal 5.0 [Durillo & Nebro, 2011], [Nebro et al., 2015]. This framework is developed in Java and is well known and widely used in the literature. Settings for the algorithms have been set according to [Jiang et al., 2014] as well as default values in jMetal [Durillo & Nebro, 2011]. Population size has been set to $N = 100$ and the stopping criterion $Max_FES = 300000$. As the competing algorithms are stochastic, their results have been averaged over 20 independent runs. As a recall, both Mo-FDA-S and Mo-FDA-D are deterministic algorithms and their results have been obtained after a single run.

As mentioned earlier, we have decided to use a set of 8 functions, 5 from the ZDT family problems and 3 from the DTLZ. The dimension set is $D = 30$ for both benchmarks. Moreover, four metrics have been chosen to fully asses the performance of each algorithm. The Hypervolume (HV), the Generational Distance (GD); the Inverted Generational Distance (IGD) and the Spread. It is important to notice that the objective of the hypervolume is to be maximized while the others need to be minimized. Moreover, to compare the results obtained by the different algorithms, we used the Friedman Rank sum method to rank the approach based on their performance on each metric and each function.

4.4.4.1 2-Objective functions

In this first section, we focus on both the ZDT and DTLZ benchmark on 2-objective functions. Results from the competing algorithms are shown on Tables C.7 to C.15. On each table the values in bold highlight the best algorithm for the given function and the given metric. In addition, Tables from C.16 to C.24 show the rank for each function and each metric. From these ranks, the importance of using multiple metrics can be seen. For instance, Mo-FDA-D is regularly ranked first on the first three metrics but last on the fourth metrics. This means that Mo-FDA-D finds good Pareto Fronts, close to the true Pareto Front but the solutions are less spread that the PS of the other algorithms. Concerning the other approach, Mo-FDA-S shows more stability over all the other metrics.

Final ranks are shown on Figure 4.11. Table C.26 shows the final rank based on the values found in Table C.25. This is also illustrated in the Figure 4.11. This data shows that Mo-FDA-D is the best algorithm on three metrics, *i.e.* the Hypervolume, the GD and IGD. However, it performs the worst on the Spread metric. This shows a lack of diversity in the Pareto Front compared to other algorithms. However, Mo-FDA-S is complementary to Mo-FDA-D where it performs well on the Hypervolume and GD and outperforms the other methods on the Spread. This means that scalarization allows finding a well Pareto Front with good diversity.

An interesting conclusion that can also be seen from Table C.26 is that, on average, Mo-FDA-S and Mo-FDA-D obtained the same rank value, *i.e.* 2.25 which ties them for first place. As mentioned, this is since Mo-FDA-D does not perform well on the Spread metric. Those conclusions can be seen in Figure 4.12 representing the Pareto Sets of our algorithms on 4 selected functions.

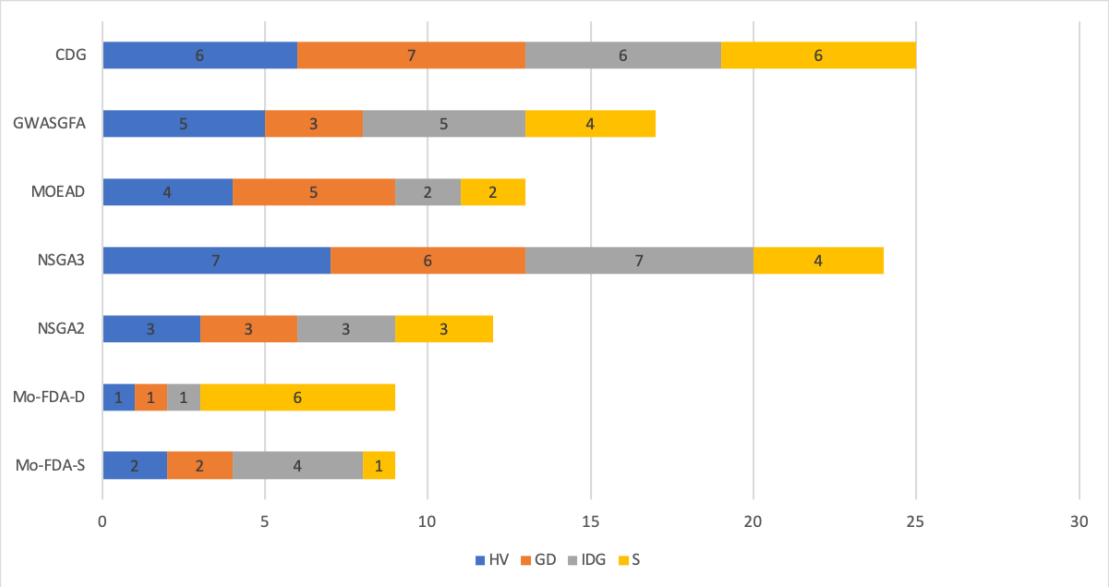


FIGURE 4.11: Illustration of the final ranking of the algorithms for 2-Objective functions for each metric.

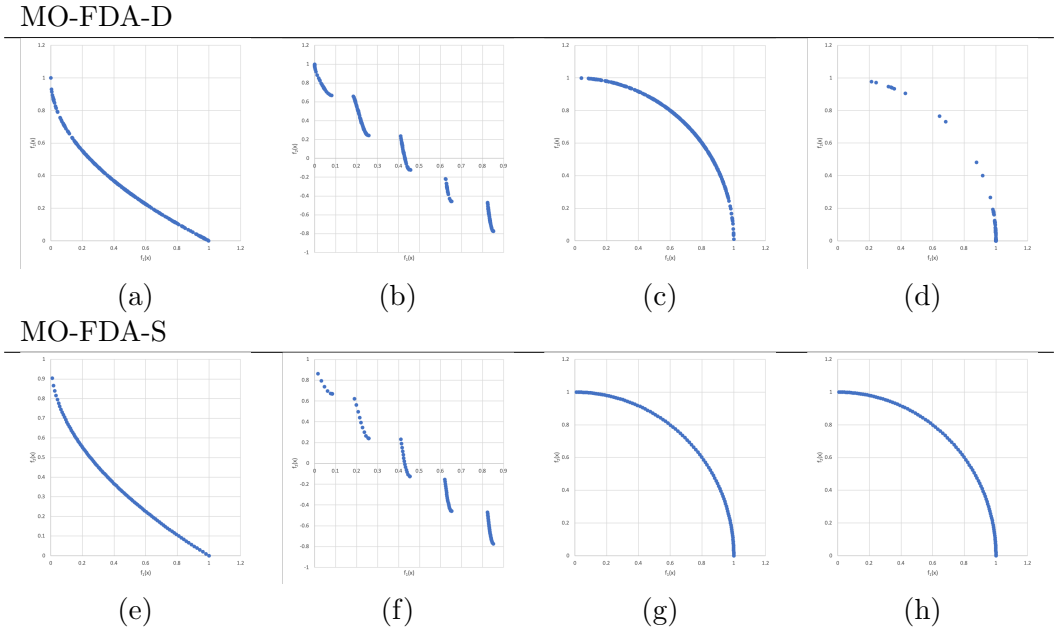


FIGURE 4.12: Pareto Fronts for four selected functions for our two approaches. With MO-FDA-D (a) ZDT1, (b) ZDT3, (c) DTLZ2, (d) DTLZ4. With MO-FDA-S (e) ZDT1, (f) ZDT3, (g) DTLZ2, (h) DTLZ4.

4.4.4.2 3-objective functions

To continue the comparison of our algorithm, we have tested Mo-FDA-D on the first four functions of the DTLZ benchmark on 3 Objectives. Indeed, ZDT is only defined for two objectives. In this context, Mo-FDA-S is not yet fit to solve 3-Objective functions. Indeed, a technique to generate well-distributed weight vectors would need to be implemented.

Results from the competing algorithms are shown in Tables C.27 to C.30. On each table the values in bold highlight the best algorithm for the given function and the given metric. In addition Tables from C.31 to C.34 show the rank for each function and each metric. Relative to the others, Mo-FDA-SD does not perform as well on 3-Objective functions. However, on each function, it outperforms the other algorithm on at least one metric.

Final ranks are shown on Figure 4.13. Table C.36 shows the final rank based on the values found in Table C.35. This is also illustrated in the Figure 4.13. This data shows that Mo-FDA-D is the best algorithm for the General Distance metric. This means that the points found by Mo-FDA-D are closer to the true Pareto Front than the other algorithm. It is ranked second on the IGD and similarly to the 2-Objective function, struggle to perform on the Spread.

The best algorithm overall is NSGA-III as it has been adapted to many-objective problems *i.e.* problems with 3 or more objective functions. This explains why it does not perform well on 2-Objective function as seen in the previous section.

Our algorithm, Mo-FDA-D is, overall, ranked second in the studied metrics, which shows promising results. Those conclusions can be seen in Figure 4.14 representing the Pareto Sets of our algorithms on four DTLZ functions with 3 objectives.

4.5 Conclusion

In this chapter, we have shown two new approaches to solve multi-objective problems. Both based on geometrical fractal decomposition using hyperspheres. The first one, Mo-FDA-S takes on the original FDA and leverages the right scalarization methods to solve MOP problems. We have combined it with a multi-node environment based on containers to allow speed increase but also architecture flexibility. The second new approach, Mo-FDA-D, has been modified at its core to use the non-dominated sorting technique during both exploration and exploitation. This is combined with an indicator based exploration using the hypervolume metric.

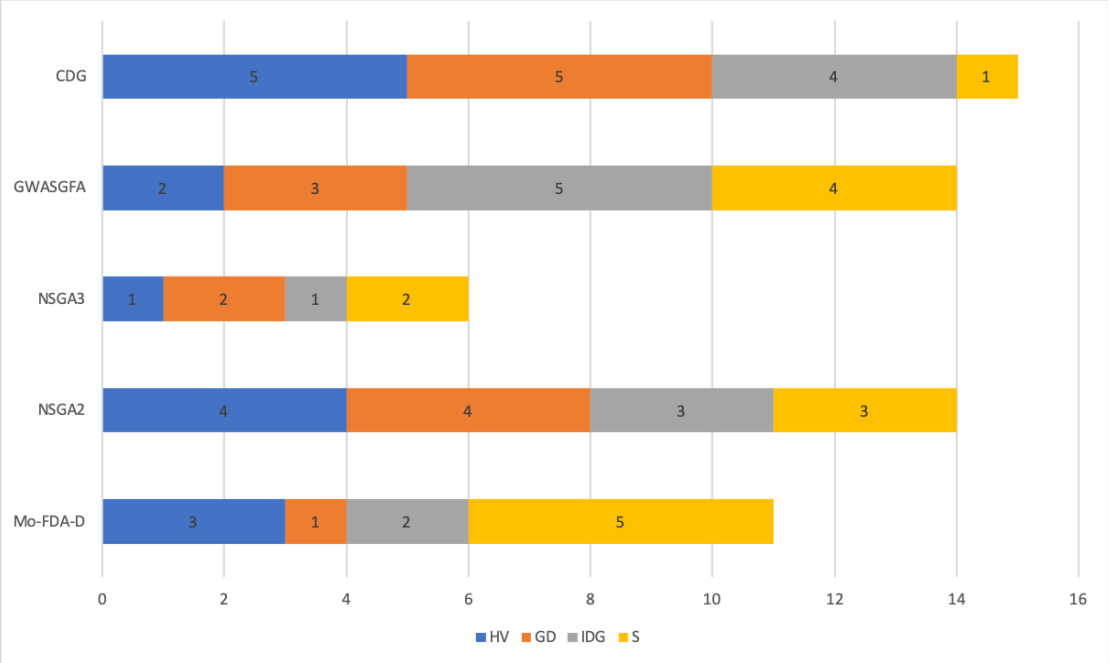


FIGURE 4.13: Illustration of the final ranking of the algorithms for 3-Objective functions for each metric.

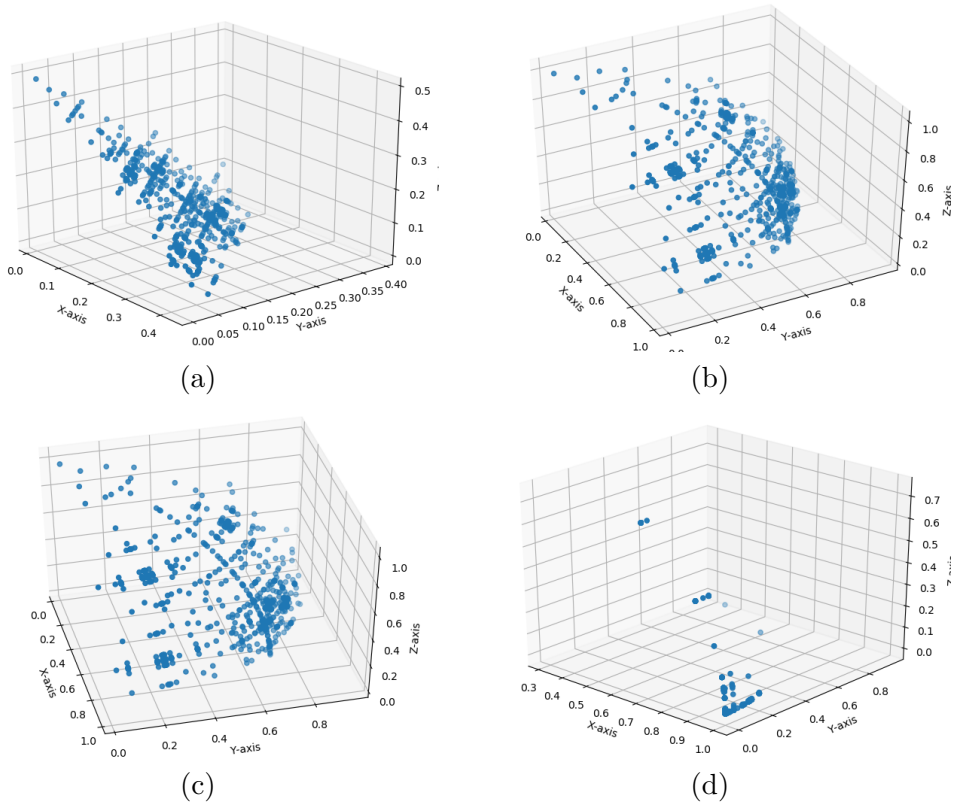


FIGURE 4.14: Pareto Fronts for the 3-Objective functions found by Mo-FDA-D. (a) DTLZ1, (b) DTLZ2, (c) DTLZ3, (d) DTLZ4.

Our algorithms have been compared to 5 other well regarded and state-of-the-art meta-heuristics. We have decided to use the four most-used metrics in the field to compare the different algorithms. It is important to note that the use of different metrics is crucial when comparing MOP methods. Indeed, each algorithm has its strengths and weaknesses and performs well on some given metrics. The use of multiple ones allows having a better overview of each algorithm. Where Mo-FDA-S performs overall well on the four metrics, Mo-FDA-D excels in finding good Pareto Front, maximizing the hypervolume covered and close to the true PF. However, it fails to find well-spread solutions.

The use of scalarization for Mo-FDA-S only works on 2-Objective functions for the moment. The other algorithm shows promising results on 3-Objective functions but, similarly to NSGA-III, could be optimized for many-objective problems.

Chapter 5

Optimal Convolution neural networks architecture search based on FDA

5.1 Introduction

Deep Learning methods [LeCun et al., 2015] have been very successful in solving difficult problems such as speech recognition, language translation or computer vision. They have been applied to any field imaginable from healthcare [Esteva et al., 2019] to computer vision for pedestrian tracking [Brunetti et al., 2018] or cybersecurity [Apruzzese et al., 2018].

This success can be attributed to the capacity of deep learning algorithms to automatically extract features from data format such as audio, image or text (known commonly as unstructured data). These technics allow shifting from manual feature engineering where engineers spend time manipulating data sets and building meaningful new features to spending time on building deep neural network architectures and optimizing their hyperparameters.

Unfortunately, training and tuning their parameters are not easy. Architectures could be composed of several layers and millions of parameters where each one needs to be optimized. In practice, neural networks are trained using simple heuristics based on gradient descent such as Stochastic Gradient Descent (SGD) [Nesterov, 1983], RMSProp [Hinton, 2012] or Adam [Kingma & Ba, 2014]. However, it might first be considered as *overkill* to try to find the optimum parameters and the computation time required to do so might be too important. Second, a neural network optimally optimized might

limit its flexibility and lead to important overfitting. Indeed, local optimum found with gradient descent would normally generalize better. Many studies have applied popular metaheuristics such as Simulated Annealing or Evolutionary Algorithms to the training of neural network architectures. However, while the accuracy increases with metaheuristics, the computation time required to train an architecture also increases [Fong et al., 2018].

Before optimizing a deep neural architectures, the task of building it has been an important focus over recent years. The complexity of the imageNet benchmark [Deng et al., 2009] has motivated engineers and scientists to push the possibilities of Deep Learning and have built deeper and more complex architectures from AlexNet [Krizhevsky et al., 2012], VGG-16 [Simonyan & Zisserman, 2014] or GoogleNet [Szegedy et al., 2015]. Building neural network architectures do result in a significant gain in performance. However, the search is very time consuming and prone to error. Metaheuristics can also be used for architectures search and fine tuning of hyperparameters. Evolutionary Algorithms have sucessfully been applied with interesting results [Real et al., 2017; Suganuma et al., 2017; Real et al., 2017; Xie & Yuille, 2017]. In this context initial architectures are defined as initial solutions and selected to create new architectures. Mutation and recombination refer to operations that lead to novel architectures in the search space. A survey on swarm and evolutionary computing applied to Deep Learning is presented in [Darwish et al., 2019].

In this chapter, we present our work in applying our approach, FDA, to the optimization of the hyperparameters of deep neural network architectures. We first present the formulated problem in Section 5.2.3. A quick review of some related work found in the literature is presented in Section 5.2.2. Our approach is then detailed in Section 5.3 followed by a discussion on our results in Section 5.4. Section 5.5 concludes this chapter.

5.2 Architecture Search and fine-tuning the hyperparameters

5.2.1 Convolution Neural Network

The basic architectural ideas of a Convolution Neural Network (CNN) [Lecun et al., 1998] consist of the local receptive fields via the convolution operation and the spatial sub-sampling via the pooling operation. The Convolution operation can be formally written as:

$$f_{x,y,k}^{C,l} = \mathbf{w}_k^{l\,T} f_{x,y}^{Op,l-1} + b_k^l \tag{5.1}$$

where $\mathbf{w}_k^l$ and b_k^l are the weights and bias of the k^{th} feature map, $f^{Op,l-1}$ and $f_{x,y,k}^{C,l}$ are the input and output feature maps, l denotes the layer and (x, y) is the spatial image coordinate. The superscript C denotes convolution and Op represents various operations, *e.g.*, input (when $l = 1$), convolution, pooling, activation, etc.

Pooling applies local operations, *e.g.*, computing the maximum within a local neighborhood has the following form:

$$f_{x,y,k}^{P_{max},l} = max_{(m,n)\in\mathcal{N}_{x,y}}(f_{m,n,k}^{Op,l-1}) \tag{5.2}$$

where, $\mathcal{N}_{x,y}$ denotes the local spatial neighborhood and $Pmax$ denotes the max pooling. Often a spatial resolution reduction is applied after the max-pooling operation. Besides the two above-mentioned operations, there are several strategies applied within the CNN models, such as non-linear activation (*e.g.*, the Rectified Linear Unit (ReLU) [He et al., 2015]), dropout [Srivastava et al., 2014] and batch normalization [Ioffe & Szegedy, 2015]. A layer with full connections, called Fully Connected (FC) layer, often appears at the end of the concatenated layers. It takes all points (neurons) from the feature maps of the previous layer as input and connects it to every points (neurons) of the output feature map.

In our case, on the last dense layer of the CNN model (referred to as the prediction layer) we used the popular Softmax activation function defined as follow:

$$\text{Softmax} = \frac{\exp(z_j)}{\sum_{k=1}^{K}\exp(z_k)} \tag{5.3}$$

where K denotes the number of training samples.

Finaly, to optimize the parameters $w.r.t$ a loss function, we used the Categorical cross-entropy loss function defined as follow:

$$\mathcal{L}_{Categoricalcross-entropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbf{1}_{y_i\in K_k}\log p_{\text{model}}\left[y_i \in K_k\right] \tag{5.4}$$

with K the number of categories and N the number of observations. The term $p_{model}[y_i \in K_k]$ is the probability predicted by the model for the i_{th} observation to belong to the k_{th} category.

5.2.2 Related Work

Several surveys are available in the literature on Neural Architecture Search (NAS). In [Elsken et al., 2018], the authors have defined NAS in three dimensions: search space, search strategy and performance estimation strategy.

In [Wistuba et al., 2019], the authors presented two methods to define the search space of the entire neural architecture, referred to as the global search space. This allows a large degree of freedom regarding the arrangement of operations within the network. It can be seen as a "*template architecture*". Its simplest example is the *chain-structure* search space and consists of an architecture represented by an arbitrary sequence of ordered nodes. [Baker et al., 2017] studied this representation of the search space by considering a fixed set of operations such as convolution, max pooling, activation and other parameters such as kernels size, stride and pooling size. They also integrated constrains when building architecture to avoid non-feasible, patently poor or computationally expensive scenarii. Another method to represent the search space is *Cell-Based*, based on the assumption that an architecture is a combination of different cells which are repeated to build the complete network. This approach has been presented in [Zoph et al., 2018], however, it requires an important computation power and the cell-based architectures are significantly more complex than the chain-structure ones.

Many strategies have been studied in the literature to explore the search space: Focused Grid search, was studied in [Pontes et al., 2016] to optimize quantitative factors of ANN design. However, both [Baker et al., 2017; Zoph et al., 2018] focused on optimizing the architecture itself and used well-known hyperparameters to run their best resulting architectures.

Lu et al. [2019] proposed the use of multi-objective optimization algorithm for cell-based architecture search. Real et al. [2017] used evolutionary algorithms to build large networks for image classification. Genetic Algorithms have been used in [Suganuma et al., 2017] and consider a wider range of operations in comparison to Real et al. [2017] and encoded their architecture as a sequence of blocks seen as the genotype of the network. In Xie & Yuille [2017], authors used an encoding method to represent CNN architecture in a fixed-length binary string and applied GA with standard genetic operations, *i.e.* selection to eliminate low performing individuals, crossover and mutation to generate new architectures. Overall EAs are performing well in the context of architecture search. However, their computational time required is very expensive as all individuals in a population representing architectures were trained. To address this issue, [Camero et al., 2019] developed a method, using the mean absolute error random sampling, to compare multiple-hidden-layer architectures. They infer the numerical accuracy of a

network without actually training it. This allows to quickly compare multiple architectures generated. However, this approach has only been applied to small architectures composed of only three hidden layers.

Less computational expensive methods were used with *Surrogate Model-Based Optimization* as in [Domhan et al., 2015] where the authors used Bayesian optimization. Indeed, training neural network requires expensive computation time on GPUs and other distributed environments. For instance, [Real et al., 2018b] reported running their architecture search using 450 GPUs. Some studies decided to train on smaller training size [Klein et al., 2016] or on small data set. For instance, in [Zoph et al., 2018], authors searched for the best cells on a smaller data set, took the best architecture found and increased the number of cells to solve a larger benchmark. Another simple technique consists of early stopping training when the training curve or the loss function does not reach a certain level after a given number of epochs [Baker et al., 2017].

Authors in Wistuba et al. [2019] concluded that a simple random search, outperforms many of the previously described methods in the search for cell-based architectures. In both [Sciuto et al., 2019] and [Li & Talwalkar, 2019], authors showed empirical results where a random search generated architectures performed at least as well as the ones obtained from established optimizers for CNNs.

5.2.3 Problem formulation

As mentioned, in the literature both architecture search and hyperparameters optimization can be found. In some studies, the search space includes both such as in [Domhan et al., 2015] and in others, only the architecture is optimized with hyperparameters fixed with values taken from the literature as in [Baker et al., 2017]. In other words, the architecture search represents the proper modelisation of the problems whereas the hyperparameters optimization is how the weights change to solve the problem. In our study, we have decided to consider both architecture search and hyperparameters optimization. Therefore, two different sets of parameters can be identified. One one hand discrete parameters related to architecture such as number of layers, number of kernel or kernel size. On the other hand, continuous parameters, related to the optimization of the neural network itself such as learning rate, weights decay. While this formulation can be applied to generate any Neural Network architecture, we decided to apply it on architecture search for Convolutional Neural Networks (CNN) for solving images classification problem. Inspired by [Sinha et al., 2014], we decided to formulate the problem as a bi-level optimization problem.

As a recall, a general formulation of bi-level optimization can be written as in Equation 5.5.

$$\begin{aligned} &\max_{\vec{x}} F(\vec{x}, \vec{y}) \\ &\text{s.t.} \\ &G(\vec{x}, \vec{y}) \leqslant 0 \\ &\max_{\vec{y}} f(\vec{x}, \vec{y}) \\ &\text{s.t.} \\ &g(\vec{x}, \vec{y}) \leqslant 0 \end{aligned} \tag{5.5}$$

with $\vec{x} \in \mathbb{R}^{n^1}$ and $\vec{y} \in \mathbb{R}^{n^2}$ are the decision variables respectively of the upper-level F with dimension n^1 and lower-level functions f with dimension n^2. Objective functions F and $f : \mathbb{R}^n \to \mathbb{R}$, $n = n^1 + n^2$.

Both objective functions F and f aim to maximize the validation accuracy defined as follow:

$$\text{Validation Accuracy} = \frac{N_{CorrectPredictions}}{N_{Total}} \tag{5.6}$$

With $N_{CorrectPredictions}$ the number of correct predictions made by the model on the validation set and N_{Total} the total number of elements in the validation set.

In our study we consider the upper-level function F, to focus on the architecture search with discrete parameters $\vec{x}$, a vector of dimension n^1, presented later in Table 5.1. The lower-level problem f optimize the continuous learning parameters $\vec{y}$, vector of dimension n^2 presented later in Table 5.3.

$G(\vec{x}, \vec{y})$ and $g(\vec{x}, \vec{y})$ are inequality constraints but in our problem, we do not have any constraints $G(\vec{x}, \vec{y})$ nor $g(\vec{x}, \vec{y})$.

5.3 Decision Variables Encoding

Herein, we propose to apply our algorithm *FDA* to solve the lower-level optimization problem where the decision variables are the hyperparameters of the model. Concerning the upper-level problem, we decided to use the random walker algorithm.

TABLE 5.1: Values range of the discrete parameters for architecture search.

Parameters	Value Range
Number of Blocks n	$\{2, 3, 4\}$
Number of Conv. Layers, k	$\{2, 3, 4\}$
Filter size, j	$\{2, 3, 4, 5\}$

TABLE 5.2: Fixed values for the architecture search.

Parameters	Values
Learning Rate	0.01
Weight Decay	10^{-4}
Learning Rate Decay	10^{-6}
Momentum	0.9
Batch Size	64
n_p Drop-out rates	$\frac{l_{th}}{2\times n_p}$
Number of filters, i	256
Number of units in the dense layers m	4000
Finale drop rate	0.2

5.3.1 Encoding of the Upper-level problem

Concerning the architecture search, we have decided to consider a chain-structure search space composed of n different blocks with each block composed of k convolution layers. Each convolution layers are composed of i number of filters, all with a L_2 regularization, a stride of 1, a $[j \times j]$ filter size, a batch normalization (as in [Ioffe & Szegedy, 2015]) and the *relu* activation function. At the end of each block, a max-pooling layer $[2 \times 2]$ and a dropout layer are added. Finally, the network ends with another batch normalization, a final dropout and a dense layer. The search space of each parameter is shown in Table 5.1 and the Algorithm 10 shows how to generate an architecture.

It is important to notice that for the number of filters i and the number of units m in the final dense layer, we considered respectively the ranges, [32,512] and [10,4000]. Those ranges are large enough for the parameters to be considered as continuous and the explicit forms of those continuous representations are indicated in the next section. However, for the purpose of the architecture search, we froze the two values to 256 filters per convolution layer and 4000 hidden units. The other continuous parameters were fixed as followed: the dropout values within each block were set as the n_{th} dropout layer, out of a total n_p dropout layers, had a dropout probability of $\frac{l_{th}}{2\times n_p}$ [Baker et al., 2017]. The dropout for the last layer is set to 20%. The Weight Decay is set to 10^{-4}. We used a Stochastic Gradient Descent (SGD) with a learning rate of 0.01, a learning rate decay of 10^{-6} and a Nesterov momentum of 0.9 with a batch size of 64. These values are summarized in Table 5.2.

With those parameters, we generated around 500 different architectures using a random search. Then, we took the best three to be optimized using FDA.

Algorithm 10: Architecture Generation

Select randomly the number of blocks n
for *Each blok* n **do**
    Select randomly the number of *2D convolution layer*, k
    for *Each convolution layer* k **do**
        Set the number of filters to 256
        Select randomly a filter size of $[j \times j]$
        Set the stride to 1
        Add a L_2 filter regularization with a weight decay
        Add a batch normalization
        Set the $RELU$ as activation function
    end
    Add a $[2 \times 2]$ MaxPooling layer
    Add a Dropout layer with a probability of $\frac{n}{2 \times n_p}$
end
Add a Drop out of 20%
Add a batch normalization
Add a dense layer with 4000 units
Add output layer with a softmax function

5.3.2 Encoding of the Lower-level problem

Once the best architectures are selected, the second phase aims to optimize the continuous parameters which are summarized in Table 5.3. They are the learning rate, weights decay, learning rate decay and momentum. We also included all dropout probabilities, *i.e.* the n_d dropout rates corresponding to the n blocks from the architecture and the final dropout. Then, four additional parameters were optimized. The first one corresponds to the amount by which the learning rate should be reduced if the validation loss was not improved after one epoch. We refer to this as *learning rate reduction on plateau*. The three others are the batch size, the number of filters i per convolutional layers and the number m of units in the final dense layer. We considered the range [32,512] for the two first parameters and [10,4000] for the number of units.

TABLE 5.3: Values range of the discrete parameters for the architecture search.

Parameters	Value Range
Learning Rate	$[0, 1]$
Weight Decay	$[0, 0.1]$
Learning Rate Decay	$[0, 0.1]$
Momentum	$[0, 1]$
Batch Size	$[32, 512]$
Learning Rate Reduction on Plateau	$[0, 1]$
n_p Drop-out rates	$[0, 1]$
Number of filters, i	$[32, 512]$
Number of units in the dense layers m	[10,4000]
Final droprate	$[0, 1]$

5.4 Results and Discussion

In this section, we present the results of our approach for the architecture search using a random walker and the hyperparameters optimization using FDA of a Convolutional Neural Network. To test our approach we used the benchmark CIFAR-10 detailed below.

The implementation was done using python as programming language and the framework Keras with Tensorflow [Abadi et al., 2016] as backend. This framework was used for its simplicity in building the neural network architecture. The experimentations were done using only three NVIDIA V100 GPUs with 16GB of RAM.

CIFAR-10

This benchmark is composed of 60000 32×32 colour images divided into ten different classes. The training set is composed of 50000 images and the test set of 10000. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The test set contains 5000 images of each class and the test set, 1000 images of each class. Classes are mutually exclusive no overlap exists between trucks and automobiles. "Automobile" only includes cars and assimilated. "Truck" includes only big trucks. Neither includes pickup trucks.

5.4.1 Optimal Architecture Search

The search was done using a random search referred to as random walker and 500 random architectures were generated. Both the chosen structure and the parameters are detailed in Section 5.3.1.

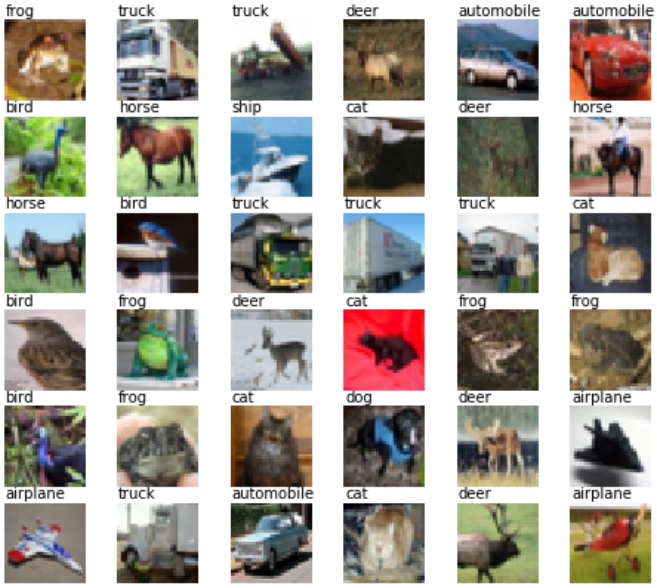


FIGURE 5.1: Examples of images in CIFAR-10 with their class labels.

As mentioned earlier, training multiple neural networks can be very expensive, this is why, to speed up the architecture search, we simplified the problem by decreasing the data set from 50000 to 20000. We used 16000 as training set and 4000 as a validation set. This allowed us to generate more architectures and still have a good estimation on the performance of each one. As the number of images was reduced, we decided to train the same architecture three times, changing the validation set each time. The final validation accuracy for one architecture was the average of the three runs. No early stopping strategy was implemented during the architecture search. The overall average performance of all architectures was 69.72% validation accuracy with a standard deviation of 10.25%. The best performing architecture reached 83.33% validation accuracy and is shown in Figure 5.2.

FIGURE 5.2: Best architecture found during the search.

5.4.2 Hyperparameter Optimization

As mentioned earlier, the 3 best architectures were selected to be optimized using FDA. To address the performance evolution mentioned in Section 5.2.2, during each training, we first decrease the learning rate if the validation loss (Equation 5.4) has not improved

after one epoch. Then, as an early stopping strategy, the training is stopped if the validation loss has not improved after two consecutive epochs.

5.4.2.1 Sensitivity analysis

We showed previously that FDA is sensitive to its fractal depth k. In this section, we conducted a sensitivity analysis of this parameter. We took the best architecture and optimized it without data augmentation to speed up the training and set k to three different values, $k = 2$, $k = 5$ and $k = 8$. Figure 5.3 shows the validation accuracy as a function of the function evaluations (trainings). The vertical ligne indicates the exploration and exploitation phases, *i.e.* the moment when ILS starts. We can see that if the fractal is deep, with $k = 8$, the accuracy is not enhanced and the ILS does not improve at all the accuracies. However, both $k = 2$ and $k = 5$ converges towards high accuracy. In the case of $k = 2$, FDA triggers ILS earlier and converges faster. In addition, the best validation accuracy reached for each value of k were 89.39% for $k = 2$, 88.12% for $k = 5$ and 49.14% for $k = 8$. Both in terms of validation accuracy and speed of convergence, the choice of $k = 2$ is the best.

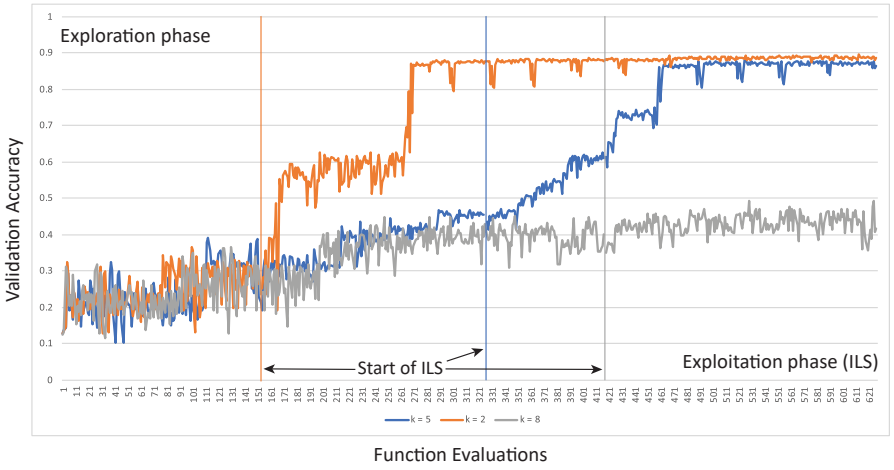


FIGURE 5.3: Validation accuracy as a function of function evaluation for each value of the parameter k.

5.4.2.2 Choice of the backpropagation algorithm

When training a neural network the choice of the backpropagation algorithm has a crucial role in the outcome. We considered adding this choice as a parameter in the architecture search but to keep consistent results, each generated architecture should be trained with all different optimizers. We decided to optimize, using FDA, the hyperparameters of the best architecture found during the search and train it with three

different optimizers *i.e.*, SGD with the Nesterov momentum [Nesterov, 1983], RMSprop and Adam. Obtained results on the 10000 images validation set were respectively 90.6% 90% and 89.5% accuracy. Besides, convergence curves are shown in Figure 5.4. As it can be seen, SGD outperforms slightly the two others. However, this is not the main reason why SGD is the best choice for hyperparameter optimizations. As in Figure 5.4, Adam and RMSprop reach quickly high accuracies during the exploration phase but results are significantly less consistent in the exploitation phase. We can empirically conclude that Adam and RMSprop do not seem to be as sensitive to optimization as SGD.

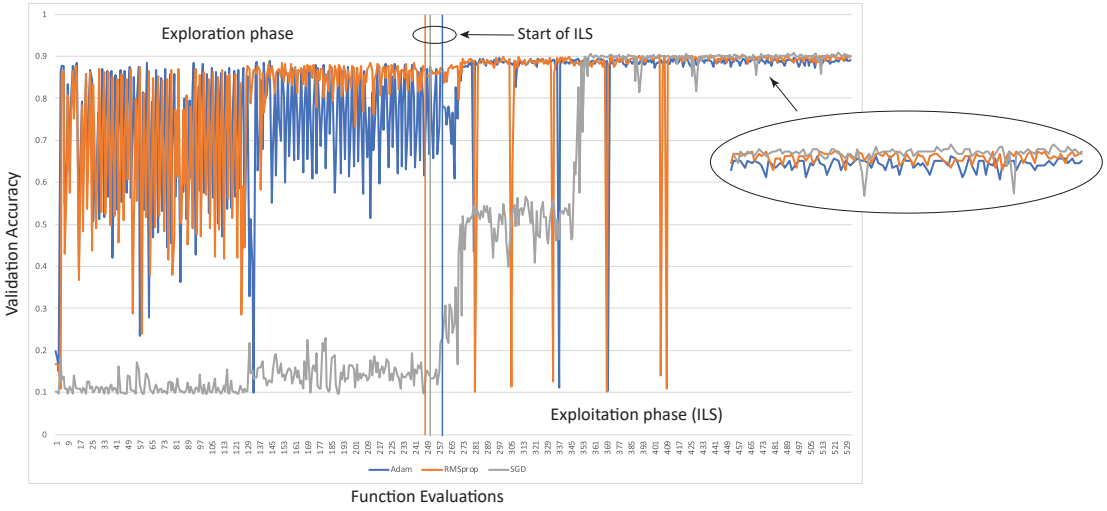


FIGURE 5.4: Comparison of three optimizers used for training our most efficient architecture.

5.4.2.3 Parameter Settings

Following the observations made while studying the sensitivity, we have found empirically that the following parameters work best: The fractal depth is set to $k = 2$ as Section 5.4.2.1. We choose the SGD for the backpropagation with the Nesterov momentum. The other FDA parameters are set as follow: The coefficient step-size $\lambda = 0.5$ and $\omega_{min} = 10^{-3}$.

The different parameters optimized have different search spaces. To facilitate the optimization of all parameters we have decided to use a normalized search space so that $X \in [0, 1]$ where X represents the set of all optimized parameters.

For the hyperparameters optimization, we used the 50000 images in the original data, 40000 were used for training and 10000 for validation with a data augmentation[1] with

[1]Data augmentation is a strategy to increase the diversity of data available for training models. It allows to generate more data without actually collecting new samples. Data augmentation transformations are applied to the original data set.

horizontal flip only, a width and height shift of 5 pixels maximum and a channel shit of 0.1. The number of epochs is set to 30.

5.4.2.4 Results

In this section, we present the results obtained for the three best architectures found during the architecture search. The best one is illustrated, before optimization, on Figure 5.2.

The challenge when optimizing an architecture was to avoid overfitting[2]. To do so we set up the following protocol. First, we optimized the parameters using only 40000 images for training and 10000 for validation. Once the best parameters are found, we run one more independent run with the same proportion but changing the validation set. This run aims to record the learning rate schedule. Then, we started ten independent runs on the 50000 images of the original training set without validation set, evaluate on the 10000 images composing the original test set and record the validation accuracy for each run. The average and standard deviation for each architecture are then reported in this section.

The best architecture described in Table 5.4 and shown on Figure (a)5.7 is composed of 4 blocks and 9 Convolution layers for 6,533,823 parameters, has an average performance on CIFAR-10 of 90.5% and a standard deviation of 0.15%. All parameters included, that represents a problem with dimension $D = 21$.

The second best architecture described in Table 5.5 and shown on Figure (b)5.7 is composed of 4 blocks with 10 Convolution Layers for 5,154,538 parameters. It has an average performance of 92.07% and a standard deviation of 0.15%. The dimension of that problem is $D = 22$.

The last optimized architecture is described in Table 5.5 and shown on Figure (c)5.7 is also composed of 4 blocks with 10 Convolution Layers and has 5,800,647 parameters. It has an average performance of 91.33% and a standard deviation of 0.22%. The dimension of that problem is $D = 22$.

The convergence graph of the 3 best architectures is shown in Figure 5.5. The second is slower to reach its plateau but ends up with better performances. The first one triggers later the exploration phase, converges fast but reaches the worst performances out of the 3 architectures studied. The three architectures had the same stopping criterion:

[2]Overfitting is defined as the production of a model that corresponds too closely to a particular training data set and hence, fail to fit additional data or predict future observations reliably

the maximum number of function evaluations, Figure 5.6 shows the second architecture which provides the best results.

When using FDA to optimize a given architecture, the best set of parameters are found. The accuracy can be seen as the upper bound value for a given architecture. This behavior can be seen on Figure 5.6. When tuning a neural network, it is difficult to know if improving the accuracy will be achieved by tuning its parameters or the architecture. Using our approach, it is clear that if one wants to improve the accuracy of an architecture, after being optimized with FDA, one should change the architecture itself or use more data augmentation.

It is interesting to the see how the optimization plateaus when reaching the best results possible. One can argue that looking at this behaviour, FDA could be used to find the upper bound of a given architecture. Indeed, FDA finds the best parameters for a given architecture. When tuning a neural network, it is difficult to know which parameter will improve the final accuracy. Using our approach, it is clear that if one wants to improve the final results, one should change the architecture itself or use more data augmentation.

To confirm the performance of our approach, we have to train our best architecture with other optimizers, RMSprop and Adam, following the same protocol. To do so we used state-of-the-art parameters taken from the literature [Nesterov, 1983; Hinton, 2012; Kingma & Ba, 2014]. Indeed, those parameters are often used in the literature as they have been tuned for a decade by researchers. Using RMSprop the architecture reaches 88.2% validation accuracy and 87.96% with Adam. That highlights the efficiency of our approach, indeed, without any particular knowledge, FDA finds the best set of parameters for a given problem than the one used in the literature. Those parameters can be found in the well-known and widely used, deep learning framework Keras. We show an improvement of around 4% to 5%.

Finally, to understand the behaviour of our best architecture and parameters, Figure 5.8 shows the learning rate as a function of the number of epochs using our reduction on plateau strategy.

5.5 Conclusion

In this chapter, we have applied FDA to the optimization of hyperparameters of Convolutional Neural Networks. We defined the problem as a bi-level optimization where the upper-level function represents the generation of the architecture itself. To solve this problem, we used a random walker and defined the architecture search space as

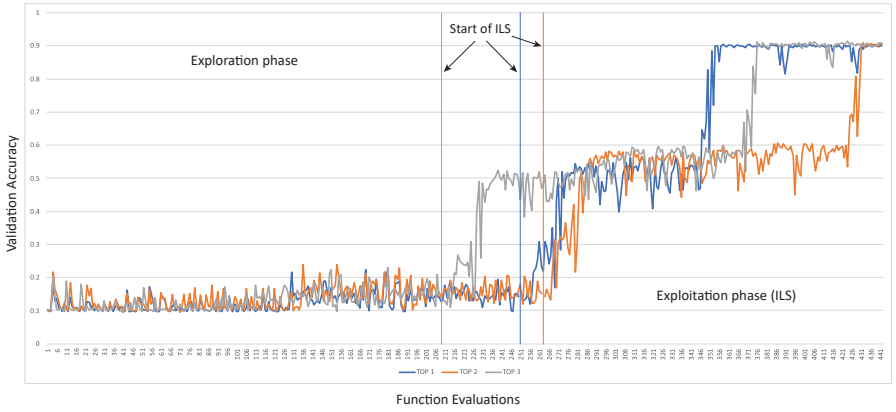


FIGURE 5.5: Comparison of the 3 best architectures convergence graph.

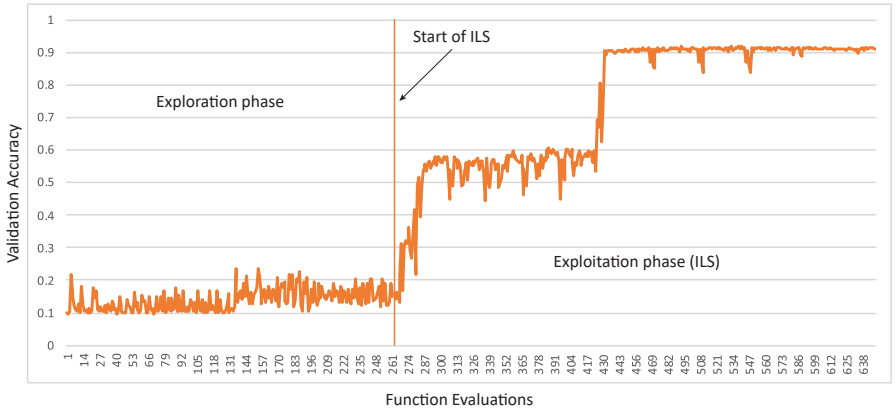


FIGURE 5.6: The second best architecture giving the best results after optimization. The vertical line highlights the beginning of ILS.

chain-structure. For the lower-level problem, we used FDA to optimize the hyperparameters of the best 3 architectures found during the random search. We defined a procedure to avoid overfitting with our parameters. The resulting architecture once optimized reached 92% validation accuracy on the 10000 images composing the test set. We compared our results against the state-of-the-art parameters used in the literature. Competitive approaches include [Baker et al., 2017] which used reinforcement learning (RL) for architecture search and reached 93.08% accuracy with 11.18 million parameters in the final neural network architecture. [Lu et al., 2019] reached 97.98% accuracy with 4 million parameters using multi-objective optimization and the well-known NSGA algorithm. Table 5.7 summarize the results FDA and other state-of-the-art methods. However, our approach differs from the others listed in Table 5.7 because it separates the architecture search from the hyperparameters optimization where the others focus on the architecture search and fine tune the hyperparameters after the search. In this study, we proved that for a given architecture FDA is capable to find better parameters and show a gain of 4 to 5%. Proportionally to the size of our neural network in terms of parameters and the data augmentation, we can argue that our results are among the

TABLE 5.4: Description of the Top 1 architecture and results on CIFAR-10.

Architecture structure		
B_1 [C(136,2,1), C(356,5,1), MaxPool(2,2), Dropout(0.237)] ; B_2 [C(304,5,1), C(171,4,1), MaxPool(2,2), Dropout(0.237)] ; B_3 [C(136,2,1), C(34,2,1), C(34,4,1), MaxPool(2,2), Dropout(0.237)] ; B_4 [C(413,3,1), C(136,2,1), MaxPool(2,2),Dropout(0.237)] ; Dropout(0.371), Dense(2329)		
Parameters		
Learning Rate	0.0057	
Weight Decay	0.0894	
Learning Rate Decay	0.0106	
Momentum	0.8941	
Batch Size	178	
Learning Rate Reduction on Plateau	0.6313	
Results on CIFAR-10		
Average validation accuracy	Std.	Nb of Parameters
90.5%	0.15%	6,533,823

TABLE 5.5: Description of the Top 2 architecture and results on CIFAR-10.

Architecture structure		
B_1 [C(212,3,1), C(304,2,1), MaxPool(2,2),Dropout(0.237)]] ; B_2 [C(136,4,1), C(212,3,1), C(413,5,1),MaxPool(2,2),Dropout(0.237)] ; B_3 [C(77,5,1), C(136,5,1),MaxPool(2,2),Dropout(0.106)]] ; B_4 [C212,2,1], C(77,2,1), C(77,2,1),MaxPool(2,2),Dropout(0.237)] ; Dropout(0.237), Dense(1597)		
Parameters		
Learning Rate	0.0057	
Weight Decay	0.0894	
Learning Rate Decay	0.0106	
Momentum	0.8941	
Batch Size	209	
Learning Rate Reduction on Plateau	0.5	
Results on CIFAR-10		
Average validation accuracy	Std	Nb Param
92.07%	0.15%	5,154,538

state-of-the-art.

TABLE 5.6: Description of the Top 3 architecture and results on CIFAR-10.

Architecture structure		
B_1 [C(304,2,1), C(304,3,1), MaxPool(2,2), Dropout(0.237)] ;		
B_2 [C(136,4,1), C(34,3,1,), C(304,4,1), C(212,2,1), MaxPool(2,2),Dropout(0.237)]		
B_3 [C(304,2,1), C(304,5,1), MaxPool(2,2), Dropout(0.237)];		
B_4 [C(136,2,1), C(304,5,1), MaxPool(2,2), Dropout(0.237)];		
Dropout(0.237), Dense(45)		
Parameters		
Learning Rate	0.0057	
Weight Decay	0.0894	
Learning Rate Decay	0.0106	
Momentum	0.8941	
Batch Size	146	
Learning Rate Reduction on Plateau	0.24	
Results on CIFAR-10		
Average validation accuracy	Std	Nb Param
91.33%	0.22%	5,800,647

TABLE 5.7: Performance of FDA and other state-of-the-art models on CIFAR-10.

Name of Architecture	**Nb of Params (M)**	**Test Error (%)**	**Search Method**
FDA	5.15	7.93	Random + FDA
MetaQNN [Baker et al., 2017]	11.18	6.92	RL
NSGA-NET [Lu et al., 2019]	4	2.02	evolution
AmoebaNet-A [Real et al., 2018a]	3.2	3.34	evolution
NAS [Zoph & Le, 2016]	7.1	4.47	RL
NAS + more filters [Zoph & Le, 2016]	37.4	4.47	RL

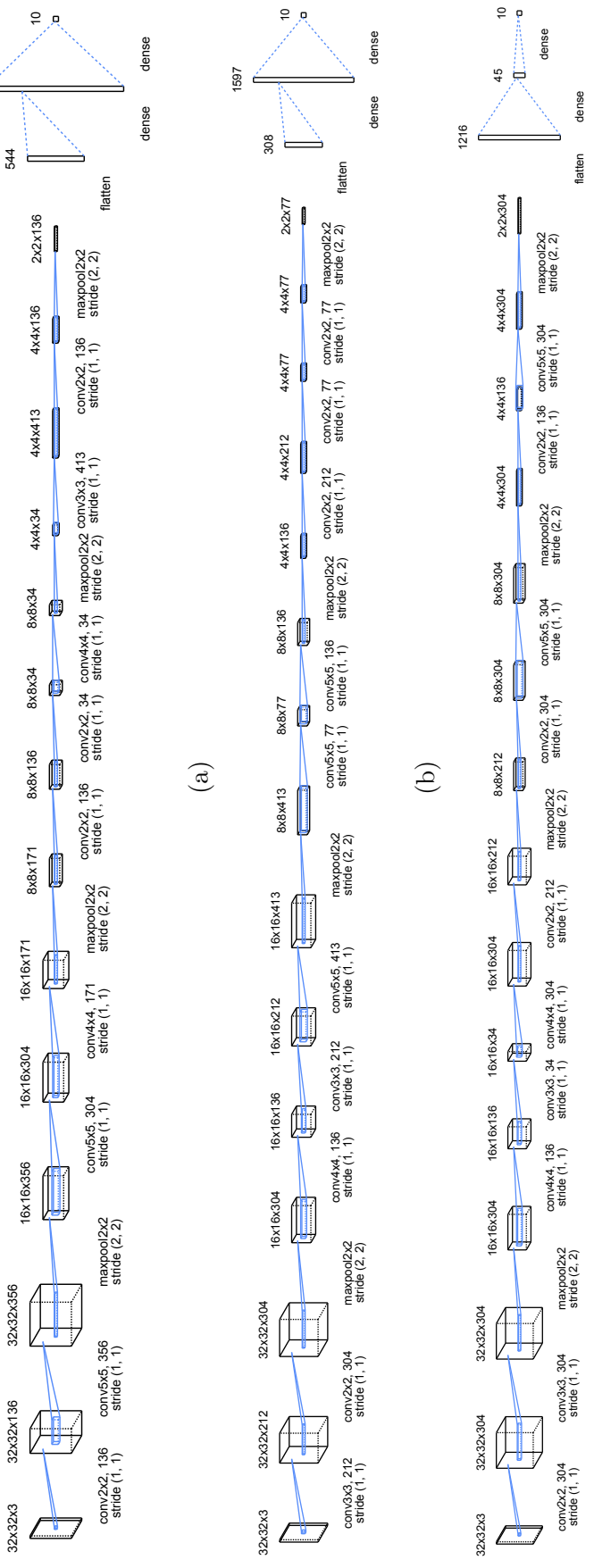


FIGURE 5.7: Top 3 Architectures after the hyperparameters optimization. (a) Top 1 with 6,533,823 parameters; (b) Top 2 with 5,154,538 parameters; (c) Top 3 with 5,800,647 parameters.

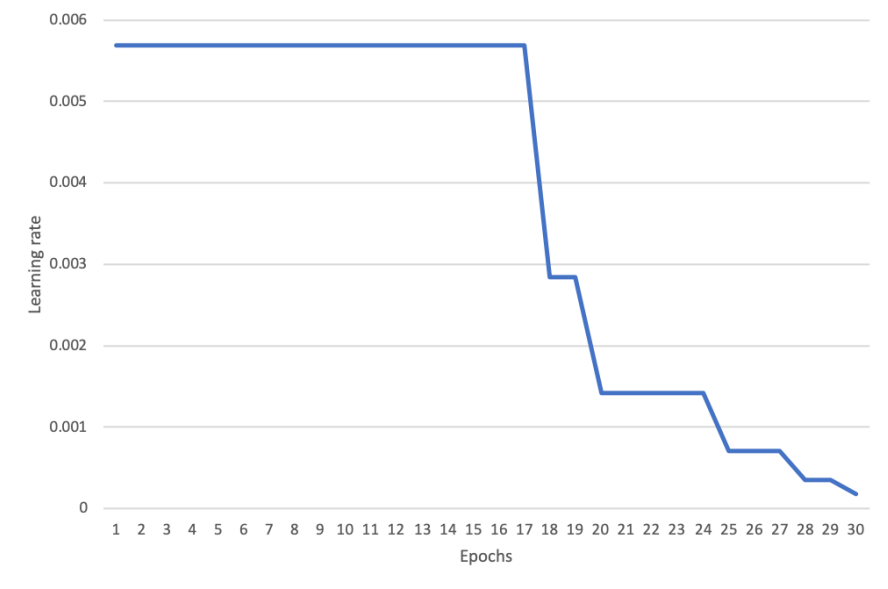


FIGURE 5.8: Learning rate as a function of the epochs for our best resulting architecture.

Conclusion and future work

In this thesis after a review of the literature, we have introduced our new metaheuristics called "Fractal Decomposition Algorithm". The approach aims to decompose the search space using hyperspheres as fractal geometrical form and can be seen as a *Divide and conquer* approach. Once the exploration phase is done a heuristic with a minimum complexity is applied to search sub-regions defined as promising to find the best solution possible. This heuristic called *Intensive Local Search* (ILS) is used during the intensification phase. Our new approach was tested on different large-scale black-box continuous functions and compared to other continuous optimization algorithms. Results showed the competitiveness in terms of accuracy and scalability of our approach.

As FDA was originally developed to run on a single thread on a single host, we studied its parallelization. Two different approaches came out of our study. One called PFDA which is designed to run on multi-threaded environments. The approach has been extensively tested on the SOCO 2011 Benchmark on large scale problems, with dimension from 50 to 5000. Using the SpeedUp as a performance metric, it is clear that PFDA is significantly faster than FDA. As PFDA needs a lot of functions evaluations (due to the parallelization) the accuracy can decrease if the stopping criterion is based on the number of function evaluations. However, if the stopping criterion is based on target precision, accuracy is maintained and high SpeedUp is obtained. It can be concluded that this new approach enforces the original strengths as it converges significantly faster. The second parallelized version, developed to run on distributed IT infrastructure is called *"Multi-Agent Fractal Decomposition Algorithm"* (MA-FDA). Two versions of MA-FDA has been developed and compared to the original version. The first, referred to as MA-FDA-S1, benefits from an extended number of function evaluations. Performance-wise, if we increase the number of function evaluation by N (equivalent to running MA-FDA-1 on N nodes), the original version is more performant but the time required increases linearly with N. This is when MA-FDA-S1 shines, having a stable computing time regardless of the number of nodes and therefore performing better if time is set as a stopping criterion. The second version, MA-FDA-S2 is designed with function evaluations as the main focus. Its running time is equivalent to the original version

but offers a better diversity and some improvement can be found when the number of nodes is not too large. From experimentations, it can be concluded that in general, both versions of MA-FDA benefit from multi-node environments

In addition to solving continuous mono-objective problems, we have shown two new approaches to solve multi-objective ones. The first one, Mo-FDA-S takes on the original FDA and leverage the Tchebycheff scalarization method. We have combined it with a multi-node environment based on containers to allow speed increase but also architecture flexibility. The second new approach, Mo-FDA-D, modifies FDA at its core to use the non-dominated sorting technique during both exploration and exploitation. This is combined with an indicator based exploration using the hypervolume metric. Our two algorithms have been compared to 5 others well regarded and state-of-the-art metaheuristics. Also, we show the interest in using the four most-used metrics to compare the different algorithms. Indeed, each algorithm has its strengths, weaknesses and performs well on some given metrics. The use of multiple metrics allows having a better overview of each algorithm. Where Mo-FDA-S performs overall well on the four metrics, Mo-FDA-D allows to find good Pareto Front maximizing the hypervolume covered and close to the true PF. However, it fails to find well-spread solutions.

We finally applied FDA to the optimization of hyperparameters of Convolutional Neural Networks for image classification problems. We defined the problem as a bi-level optimization where the upper-level function represents the generation of the neural network architecture. A random walker was used to find a good chain-structure architecture. For the lower-level function, we used FDA to optimize the hyperparameters of the best 3 architecture found during the random search. Indeed, fine-tuning manually a neural network architecture is a very time-consuming task. One challenge we faced was avoiding overfitting when optimizing the parameters. To test the performance of our approach we used a popular benchmark, the CIFAR-10. It is composed of 50000 images as training set and 10000 images as test set. Our approach FDA found hyperparameters which, after 10 independent runs, reached 92% validation accuracy on the test set. In this study, we proved that for a given architecture FDA is capable of finding better parameters and show a gain of 4 to 5% which is an important gain in this field. Proportionally to the size of our neural network in terms of parameters and the data augmentation, we can argue that our results are among the state-of-the-art. This was done with only three Nvidia V100 GPUs wherein many other studies on architecture search, infrastructure with hundreds of GPUs are used.

While FDA has shown its potential on multiple functions, the heuristics used during the intensification phase referred to as *ILS* has some limitations. As a recall, ILS moves along each dimension one by one. This process can increase the difficulty of solving

non-separable problems. Various techniques could be used to address this issue. One solution could be to hybridize FDA with another metaheuristic. For instance, the well-known CMA-ES could be used instead of ILS. In small-scale problem, this could improve the accuracy on problems with small dimensions. Any heuristics or metaheuristics could be used instead of ILS and could be chosen according to the dimension of the problem. Another solution would be to adapt ILS itself. For instance, integrating a clustering algorithm to group decision variables that behave similarly. Therefore instead of changing one dimension at a time, ILS move along each cluster.

Another improvement concerning FDA would be to explore other strategies to evaluate hyperspheres. For instance, points evenly (or normally) distributed on the hypersphere could be selected. Hypersphere selection strategies are an interesting lead to investigate in order to improve FDA.

Regarding the parallelized versions, FDA, MA-FDA-1 and MA-FDA-2, all suffer from the same problem in solving non-separable functions. To improve FDA, it will be important to take into account parallelization possibilities and constraints. However, on the distributed approaches, parallelization could go even further in running the exploration and exploitation phases on Graphics Processing Units (GPU).

About the multi-objective versions of FDA, Mo-FDA-S only works now on 2-Objective. This is due to the way the weights are calculated. The literature shows different techniques to compute weights vectors for 3 and more objective functions. The dominance-based version of Mo-FDA shows interesting results on both 2-Objective and 3-Objective problems. However, a solution to improve the spread of solutions found could be studied. For instance, Mo-FDA-D uses the hypervolume as a selection indicator for the hyperspheres. The Spread metric could also be used to select the best solutions to search around using ILS. After improving the spread of solutions, the algorithm should be tested on many-objective benchmarks.

Finally, different leads could be followed to continue our application on optimizing hyperparameters in a convolutional neural network. Rethinking the architecture search and include additional features that are known to improve performance such as skip connection or other types of operations. A cell-based search space could also be studied instead of the chain-structure that we used. Deeper architectures could also be studied with more parameters. Other leads to study could be to apply FDA to other types of Neural Networks such as LSTM or RNN for other tasks such as text translation or voice recognition.

Appendix A

Tables of chapter 2 - FDA

This appendix contains all tables regarding the Chapter 2

TABLE A.1: Functions F_1-F_{11}

Function	Name	Definition
F1	Shifted Sphere Function	$\sum_{i=1}^{D} z_i^2 + f\_bias, z = x - o$
F2	Shifted Schwefel Problem 2.21	$\max\{\|z_i\|, 1 \leq i \leq D\} + f\_bias, z = x - o$
F3	Shifted Rosenbrock's Function	$\sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1^2)) + f\_bias, z = x - o$
F4	Shifted Rastrigin's Function	$\sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias, z = x - o$
F5	Shifted Griewank's Function	$\sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{z_i}{\sqrt{i}}) + 1 + f\_bias, z = x - o$
F6	Shifted Ackley's Function	$-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e + f\_bias, z = x - o$
F7	Schwefel's Problem 2.22	$\sum_{i=1}^{D} \|x_i\| + \prod_{i=1}^{D} \|x_i\|$
F8	Schwefel's Problem 1.2	$\sum_{i=1}^{D}(\sum_{j=1}^{D} x_j)^2$
F9	Extended f10	$\sum_{i=1}^{D-1} f_{10}(x_i, x_{i+1})) + f_{10}(x_D, x_1)$ $f_{10}(x, y) = (x^2 + y^2)^{0.25}(sin^2(50(x^2 + y^2)^{0.1}) + 1)$
F10	Bohachevsky	$\sum_{i=1}^{D-1}(x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7)$
F11	Schaffer	$(x_i^2 + x_{i+1}^2)^{0.25}(sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$

TABLE A.2: Properties of functions F_1-F_{11}

Function	Range	Optimum	U/M	Shifted	Separable	Can be optimized dimension by dimension
F1	$[-100, 100]^D$	-450	U	✓	✓	✓
F2	$[-100, 100]^D$	-450	U	✓		
F3	$[-100, 100]^D$	390	M	✓		✓
F4	$[-5, 5]^D$	-330	M	✓	✓	✓
F5	$[-600, 600]^D$	-180	M	✓		
F6	$[-32, 32]^D$	-140	M	✓	✓	✓
F7	$[-10, 10]^D$	0	U		✓	✓
F8	$[-65.536, 65.536]^D$	0	U			
F9	$[-100, 100]^D$	0	U			✓
F10	$[-15, 15]^D$	0	U			
F11	$[-100, 100]^D$	0	U			

TABLE A.3: Properties of functions F_{12}-F_{19}

Function	F_{ns}	F'	m_{ns}	Range	$f(x^*)$
F12	$NS - F_9$	F_1	0.25	$[-100, 100]^D$	0
F13	$NS - F_9$	F_3	0.25	$[-100, 100]^D$	0
F14	$NS - F_9$	F_4	0.25	$[-5, 5]^D$	0
F15	$NS - F_{10}$	$NS - F_7$	0.25	$[-10, 10]^D$	0
F16	$NS - F_9$	F_1	0.75	$[-100, 100]^D$	0
F17	$NS - F_9$	F_3	0.75	$[-100, 100]^D$	0
F18	$NS - F_9$	F_4	0.75	$[-5, 5]^D$	0
F19	$NS - F_{10}$	$NS - F_7$	0.75	$[-10, 10]^D$	0

TABLE A.4: Sensitivity analysis with respect to the fractal depth (k). The average error is computed for $D = 200$, $D = 500$ and $D = 1000$.

	D = 200				D = 500				D = 1000			
k	3	4	5	6	3	4	5	6	3	4	5	6
$F2$	**5.39E-11**	6.44E-11	1.23E-10	1.31E-10	**1.43E-04**	1.85E-04	4.30E-04	1.23E-03	**1.28E-01**	1.56E-01	3.11E-01	9.28E-01
$F3$	1.75E+03	7.58E+02	**2.51E+02**	8.34E+03	8.55E+02	5.33E+02	5.82E+02	5.30E+02	1.14E+03	**9.37E+02**	1.13E+03	1.15E+03
$F4$	1.30E+02	7.49E+02	**0.00E+00**	1.50E+03	3.50E+02	2.02E+03	**0.00E+00**	3.55E+03	7.12E+02	4.28E+03	0.00E+00	7.02E+03
$F5$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
$F6$	3.06E-13	3.27E-13	**2.52E-13**	1.90E+01	9.20E-13	**7.85E-13**	8.74E-13	1.90E+01	1.78E-12	**1.71E-12**	1.91E-12	1.92E+01
$F13$	1.63E+02	1.18E+02	**7.07E+01**	2.19E+02	4.07E+02	8.30E+02	**3.74E+02**	4.11E+02	2.07E+04	**6.99E+02**	7.69E+02	8.53E+02
$F14$	9.85E+01	5.49E+02	**0.00E+00**	1.20E+03	2.65E+02	1.43E+03	**0.00E+00**	2.65E+03	5.32E+02	3.13E+03	**0.00E+00**	5.16E+03
$F17$	**6.06E+01**	4.26E+03	9.31E+01	8.13E+01	**1.46E+02**	5.46E+03	3.96E+02	1.55E+02	2.33E+03	2.74E+02	**1.95E+02**	2.81E+02
$F18$	2.79E+01	1.97E+02	**0.00E+00**	4.71E+02	8.16E+01	4.70E+02	**0.00E+00**	1.00E+03	1.67E+02	9.35E+02	**0.00E+00**	1.81E+03

TABLE A.5: Complexity of methods of the FDA.

Step	Asymptotic complexity
Fractal decomposition	$\log_k(D)$
Quality evaluation of a hypersphere	1
ILS	$\log_2(r/\alpha_{min})$

TABLE A.6: Experimental results obtained by FDA on functions $F_1 - F_7$

Dimension	F_1	F_2	F_3	F_4	F_5	F_6	F_7
$50D$	$0.0000E{+}00$	$2.71E-12$	$9.32E+01$	$0.00E+00$	$0.00E+00$	$6.75E-14$	$0.00E+00$
$100D$	$0.0000E{+}00$	$8.48E-12$	$5.09E+01$	$0.00E+00$	$0.00E+00$	$1.35E-13$	$0.00E+00$
$200D$	$0.0000E{+}00$	$1.23E-10$	$2.51E+02$	$0.00E+00$	$0.00E+00$	$2.52E-13$	$0.00E+00$
$500D$	$0.0000E{+}00$	$4.30E-04$	$5.82E+02$	$0.00E+00$	$0.00E+00$	$8.74E-13$	$0.00E+00$
$1000D$	$0.0000E{+}00$	$3.11E-01$	$1.13E+03$	$0.00E+00$	$0.00E+00$	$1.91E-12$	$0.00E+00$

TABLE A.7: Experimental results obtained by FDA on functions $F_8 - F_{14}$

Dimension	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
$50D$	$0.0000E{+}00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$5.50E+01$	$0.00E+00$
$100D$	$0.0000E{+}00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.68E+02$	$0.00E+00$
$200D$	$0.0000E{+}00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$7.07E+01$	$0.00E+00$
$500D$	$0.0000E{+}00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$3.74E+02$	$0.00E+00$
$1000D$	$0.0000E{+}00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$7.69E+02$	$0.00E+00$

TABLE A.8: Experimental results obtained by FDA on functions $F_{15} - F_{19}$

Dimension	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
$50D$	$0.00E+00$	$0.00E+00$	$6.09E-04$	$0.00E+00$	$0.00E+00$
$100D$	$0.00E+00$	$0.00E+00$	$6.22E+00$	$0.00E+00$	$0.00E+00$
$200D$	$0.00E+00$	$0.00E+00$	$9.31E+01$	$0.00E+00$	$0.00E+00$
$500D$	$0.00E+00$	$0.00E+00$	$3.96E+02$	$0.00E+00$	$0.00E+00$
$1000D$	$0.00E+00$	$0.00E+00$	$1.95E+02$	$0.00E+00$	$0.00E+00$

TABLE A.9: Number of evaluations to find the best solution for F_3, F_4 and F_{16} for dimensions $D = 50$ and $D = 1000$

	$D = 50$	$D = 1000$
F_3	250000	5000000
F_4	7802	160002
F_{16}	12802	268002

TABLE A.10: Number of spheres visisted for F_3, F_4 and F_{16} for dimensions $D = 50$ and $D = 1000$

	$D = 50$	$D = 1000$
F_3	5	5
F_4	29	29
F_{16}	23	23

TABLE A.11: Comparison of FDA and DIRECT algorithms for dimensions $D = 50$ and $D = 100$

Dimensions	D = 50		D = 100	
	DIRECT	FDA	DIRECT	FDA
$F1$	5.53E+01	**0.00E+00**	1.06E+04	**0.00E+00**
$F2$	5.53E+01	**2.71E-12**	7.45E+01	**8.48E-12**
$F3$	1.79E+04	**9.32E+01**	9.84E+07	**5.09E+01**
$F4$	1.26E+02	**0.00E+00**	6.64E+02	**0.00E+00**
$F5$	1.06E+00	**0.00E+00**	5.06E+01	**0.00E+00**
$F6$	1.59E-01	**6.75E-14**	1.87E-01	**1.35E-13**

Values in bold represent the best value found between DIRECT and FDA

TABLE A.12: The complexity of algorithms being used in the comparison

Algorithm	Complexity
DE	$\mathcal{O}(D^2)$
CHC	$\mathcal{O}(D^2)$
MTS-LS1	Less than or equal to $\mathcal{O}(D^2)$
SaDE	Less than or equal to $\mathcal{O}(D^3)$
mDE-bES	Less than or equal to $\mathcal{O}(D^2)$
jDElscop	Less than or equal to $\mathcal{O}(D^2)$
MA-SSW-Chains	Greater than or equal to $\mathcal{O}(D^3)$
FDA	Less than or equal to $\mathcal{O}(\log_k(D))$

TABLE A.13: Ranking using Friedman Rank sum of all algorithms at dimensions $D = 50$, $D = 100$, $D = 200$, $D = 500$ and $D = 1000$.
(Values in parenthesis represent the algorithm's rank for the given dimension relative to the others)

Ranks for dimensions:	$D = 50$	$D = 100$	$D = 200$	$D = 500$	$D = 1000$
MA-SSW-Chains	3.92 (4)	3.74 (4)	3.97 (4)	4.68 (4)	4.84 (5)
jDElscop	3.00 (2)	2.82 (2)	2.79 (3)	2.61 (3)	2.55 (2)
CHC	6.84 (7)	7.05 (8)	7.24 (8)	7.53 (8)	7.47 (8)
mDE-bES	3.32 (3)	3.03 (3)	2.55 (2)	2.47 (2)	2.58 (3)
DE	4.63 (6)	4.82 (5)	5.50 (6)	5.58 (6)	5.97 (6)
MTS-LS1	4.58 (5)	5.29 (6)	5.13 (5)	4.87 (5)	4.39 (4)
SaDE	6.89 (8)	6.89 (7)	6.45 (7)	6.00 (7)	6.08 (7)
FDA	2.82 (1)	2.37 (1)	2.37 (1)	2.26 (1)	2.11 (1)

TABLE A.14: Raw and adjusted (using the Holm procedure) p-values from Wilcoxon test. [a] p-value >0.05, failing to show statistical difference with significan level $\alpha = 0.05$.

FDA vs.	$D = 50$		$D = 100$		$D = 200$	
	Raw	Adjusted	Raw	Adjusted	Raw	Adjusted
MA.SSW.Chains	1.62E-02	4.87E-02	2.63E-03	7.90E-03	8.19E-04	2.46E-03
jDElscop	**6.03E-01**[a]	**6.03E-01**[a]	**1.07E-01**[a]	**2.14E-01**[a]	**1.23E-01**[a]	**2.46E-01**[a]
CHC	4.10E-07	2.87E-06	1.25E-07	8.72E-07	1.26E-07	8.82E-07
mDE.bES	**2.56E-01**[a]	**5.12E-01**[a]	**2.16E-01**[a]	**2.16E-01**[a]	**5.64E-01**[a]	**5.64E-01**[a]
DE	4.45E-03	2.22E-02	6.70E-06	3.35E-05	7.48E-06	3.74E-05
MTS.LS1	7.02E-03	2.81E-02	3.18E-05	1.27E-04	5.27E-05	2.11E-04
SaDE	4.62E-07	2.87E-06	1.64E-07	9.81E-07	3.17E-07	1.90E-06

[a] p-value >0.05, failing to show statistical difference with significan level $\alpha = 0.05$.

TABLE A.15: Raw and adjusted (using the Holm procedure) p-values from Wilcoxon test

FDA vs.	$D = 500$		$D = 1000$	
	Raw	Adjusted	Raw	Adjusted
MA.SSW.Chains	4.88E-05	1.95E-04	1.23E-06	4.90E-06
jDElscop	**1.22E-01**[a]	**2.44E-01**[a]	**8.18E-02**[a]	**1.64E-01**[a]
CHC	7.98E-08	5.58E-07	8.54E-08	5.98E-07
mDE.bES	**4.93E-01**[a]	**4.93E-01**[a]	**2.48E-01**[a]	**2.48E-01**[a]
DE	3.59E-06	1.79E-05	3.30E-07	1.65E-06
MTS.LS1	3.90E-04	1.17E-03	4.97E-04	1.49E-03
SaDE	4.23E-07	2.54E-06	1.68E-07	1.01E-06

[a] p-value >0.05, failing to show statistical difference with significan level $\alpha = 0.05$.

TABLE A.16: Number of times global optimum is reached.
(Values in parenthesis represent the algorithm's rank relative to the others among all dimensions)

Dimensions	$D = 50$	$D = 100$	$D = 200$	$D = 500$	$D = 1000$	Average	Rank
MA-SSW-Chains	9	8	6	3	2	5.6	(4)
jDElscop	12	10	9	8	7	9.2	(3)
CHC	0	0	0	0	0	0	(8)
mDE-bES	9	9	11	9	9	9.4	(2)
DE	7	4	2	1	1	3	(6)
MTS-LS1	7	4	4	2	4	4.2	(5)
SaDE	1	1	1	1	1	1	(7)
FDA	14	14	14	14	14	14	(1)

TABLE A.17: Ranking using Friedman Rank sum with other algorithms at dimension $D = 50$.
Values in parenthesis represent the algorithm's rank for the given dimension relative to the others

FDA vs.	D=50	Rank
MOS-SOCO2011	2.578947368	(3)
MOS-CEC2013	4.421052632	(5)
MOS-CEC2012	5.236842105	(6)
IACO$_{\mathbb{R}}$-Hybrid	2.236842105	(1)
2S-Ensemble	4.157894737	(4)
FDA	2.368421053	(2)

TABLE A.18: Raw and adjusted (using the Holm procedure) p-values from Wilcoxon test with other metheuristics
[a] p-value >0.05, failing to show statistical difference with significan level $\alpha = 0.05$.

FDA vs.	$D = 50$	
	Raw	Adjusted
MOS-SOCO2011	**2.54E-01**[a]	**5.09E-01**[a]
MOS-CEC2013	2.12E-05	8.46E-05
MOS-CEC2012	4.58E-07	2.29E-06
IACO$_{\mathbb{R}}$-Hybrid	**9.16E-01**[a]	**9.16E-01**[a]
2S-Ensemble	5.14E-05	1.54E-04

TABLE A.19: Average error on 50D functions.

	MA-SSW-Chains	jDElscop	CHC	mDE-bES	DE	MTS-LS1	SaDE	FDA
F_1	$0.00E+00$	$0.00E+00$	$1.67E-11$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$2.68E+01$	$0.00E+00$
F_2	$7.61E-02$	$3.15E-02$	$6.19E+01$	$1.52E+01$	$8.84E-11$	$8.84E-14$	$1.21E+02$	$2.71E-12$
F_3	$4.79E+01$	$2.28E+01$	$1.25E+06$	$4.76E-05$	$1.63E+02$	$1.63E+02$	$7.46E+04$	$9.32E+01$
F_4	$1.19E-01$	$0.00E+00$	$7.43E+01$	$1.77E+01$	$0.00E+00$	$0.00E+00$	$1.07E+01$	$0.00E+00$
F_5	$0.00E+00$	$0.00E+00$	$1.67E-03$	$0.00E+00$	$7.68E-03$	$7.68E-03$	$1.87E-01$	$0.00E+00$
F_6	$4.89E-14$	$9.55E-14$	$6.15E-07$	$3.97E-14$	$0.00E+00$	$0.00E+00$	$4.63E-02$	$6.75E-14$
F_7	$0.00E+00$	$0.00E+00$	$2.66E-09$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
F_8	$3.06E-01$	$9.97E-03$	$2.24E+02$	$1.64E-09$	$9.56E-12$	$9.65E-12$	$6.92E+05$	$0.00E+00$
F_9	$2.94E+02$	$0.00E+00$	$3.10E+02$	$0.00E+00$	$1.03E+02$	$1.03E+02$	$3.00E-02$	$0.00E+00$
F_{10}	$0.00E+00$	$0.00E+00$	$7.30E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$2.94E-02$	$0.00E+00$
F_{11}	$4.49E-03$	$0.00E+00$	$2.16E+00$	$1.15E-08$	$1.04E+02$	$1.04E+02$	$8.35E-02$	$0.00E+00$
F_{12}	$0.00E+00$	$0.00E+00$	$9.57E-01$	$0.00E+00$	$1.34E+01$	$1.34E+01$	$4.80E+01$	$0.00E+00$
F_{13}	$3.02E+01$	$1.36E+01$	$2.08E+06$	$2.50E-01$	$2.94E+01$	$2.94E+01$	$3.42E+09$	$5.50E+01$
F_{14}	$0.00E+00$	$0.00E+00$	$6.17E+01$	$9.60E+00$	$5.52E+01$	$5.52E+01$	$4.22E+03$	$0.00E+00$
F_{15}	$0.00E+00$	$0.00E+00$	$3.98E-01$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$8.50E-03$	$0.00E+00$
F_{16}	$4.06E-03$	$0.00E+00$	$2.95E-09$	$0.00E+00$	$4.06E+01$	$4.06E+01$	$1.36E+01$	$0.00E+00$
F_{17}	$2.60E+01$	$7.43E-03$	$2.26E+04$	$2.42E-01$	$2.17E+02$	$2.17E+02$	$2.36E+05$	$6.09E-04$
F_{18}	$0.00E+00$	$2.41E-14$	$1.58E+01$	$5.65E-05$	$5.65E+01$	$5.65E+01$	$2.72E+01$	$0.00E+00$
F_{19}	$0.00E+00$	$0.00E+00$	$3.59E+02$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.15E-01$	$0.00E+00$

TABLE A.20: Average error on 100D functions.

	MA-SSW-Chains	jDElscop	CHC	mDE-bES	DE	MTS-LS1	SaDE	FDA
F_1	$0.00E+00$	$0.00E+00$	$3.56E-11$	$0.00E+00$	$3.79E+00$	$1.09E-12$	$3.13E+01$	$0.00E+00$
F_2	$7.01E+00$	$1.21E+00$	$8.58E+01$	$4.00E+01$	$7.58E+01$	$4.66E-10$	$1.26E+02$	$8.48E-12$
F_3	$1.38E+02$	$6.13E+01$	$4.19E+06$	$4.90E-01$	$1.27E+02$	$2.32E+02$	$1.11E+05$	$5.09E+01$
F_4	$1.19E-01$	$0.00E+00$	$2.19E+02$	$1.87E+01$	$2.85E+00$	$1.05E-12$	$1.58E+01$	$0.00E+00$
F_5	$0.00E+00$	$0.00E+00$	$3.83E-03$	$0.00E+00$	$3.05E-01$	$6.70E-03$	$3.53E-01$	$0.00E+00$
F_6	$6.03E-14$	$2.00E-13$	$4.10E-07$	$1.44E-13$	$4.34E-01$	$1.20E-12$	$8.32E-02$	$1.35E-13$
F_7	$0.00E+00$	$0.00E+00$	$1.40E-02$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
F_8	$3.48E+01$	$5.57E+00$	$1.69E+03$	$2.32E-03$	$4.74E+02$	$1.43E-03$	$2.83E+05$	$0.00E+00$
F_9	$5.63E+02$	$7.18E-09$	$5.86E+02$	$0.00E+00$	$3.71E-03$	$2.20E+02$	$3.00E-02$	$0.00E+00$
F_{10}	$0.00E+00$	$0.00E+00$	$3.30E+01$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$4.73E-02$	$0.00E+00$
F_{11}	$1.09E-01$	$8.17E-09$	$7.32E+01$	$0.00E+00$	$8.58E-04$	$2.10E+02$	$3.05E-01$	$0.00E+00$
F_{12}	$3.28E-03$	$0.00E+00$	$1.03E+01$	$5.36E-04$	$2.71E+00$	$3.91E+01$	$3.79E+01$	$0.00E+00$
F_{13}	$8.35E+01$	$5.11E+01$	$2.70E+06$	$8.50E+00$	$5.87E+01$	$1.75E+02$	$3.42E+09$	$1.68E+02$
F_{14}	$0.00E+00$	$0.00E+00$	$1.66E+02$	$1.16E+01$	$2.21E+00$	$2.04E+02$	$3.92E+03$	$0.00E+00$
F_{15}	$0.00E+00$	$0.00E+00$	$8.13E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$3.99E-02$	$0.00E+00$
F_{16}	$1.61E-02$	$0.00E+00$	$2.23E+01$	$0.00E+00$	$3.52E+00$	$1.04E+02$	$1.96E+01$	$0.00E+00$
F_{17}	$9.92E+01$	$3.21E-01$	$1.47E+05$	$6.65E-03$	$1.58E+01$	$4.17E+02$	$2.34E+05$	$6.22E+00$
F_{18}	$0.00E+00$	$6.33E-14$	$7.00E+01$	$4.46E-01$	$8.76E-01$	$1.22E+02$	$3.05E+01$	$0.00E+00$
F_{19}	$0.00E+00$	$0.00E+00$	$5.45E+02$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$2.71E-01$	$0.00E+00$

TABLE A.21: Average error on 200D functions.

	MA-SSW-Chains	jDElscop	CHC	mDE-bES	DE	MTS-LS1	SaDE	FDA
F_1	$0.00E+00$	$0.00E+00$	$8.34E-01$	$0.00E+00$	$8.55E+00$	$2.29E+00$	$2.03E+01$	$0.00E+00$
F_2	$3.36E+01$	$7.54E+00$	$1.03E+02$	$4.15E+01$	$1.05E+02$	$4.54E-09$	$1.03E+02$	$1.23E-10$
F_3	$2.50E+02$	$1.40E+02$	$2.01E+07$	$1.35E+02$	$3.32E+05$	$1.69E+02$	$4.82E+04$	$2.51E+02$
F_4	$4.43E+00$	$0.00E+00$	$5.40E+02$	$9.27E-13$	$6.98E+00$	$2.34E-12$	$6.25E+00$	$0.00E+00$
F_5	$0.00E+00$	$0.00E+00$	$8.76E-03$	$0.00E+00$	$4.05E-01$	$5.42E-03$	$6.43E-02$	$0.00E+00$
F_6	$1.19E-13$	$4.52E-13$	$1.23E+00$	$0.00E+00$	$7.14E-01$	$2.38E-12$	$2.73E-02$	$2.52E-13$
F_7	$0.00E+00$	$0.00E+00$	$2.59E-01$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
F_8	$7.23E+02$	$2.52E+02$	$9.38E+03$	$8.71E-01$	$5.76E+03$	$1.42E+01$	$4.47E+05$	$0.00E+00$
F_9	$1.17E+03$	$4.30E-08$	$1.19E+03$	$0.00E+00$	$8.79E-03$	$4.27E+02$	$3.00E-02$	$0.00E+00$
F_{10}	$0.00E+00$	$0.00E+00$	$7.13E+01$	$0.00E+00$	$4.19E-02$	$0.00E+00$	$1.59E-02$	$0.00E+00$
F_{11}	$3.50E-01$	$9.58E-09$	$3.85E+02$	$0.00E+00$	$5.07E-03$	$4.28E+02$	$4.89E-03$	$0.00E+00$
F_{12}	$1.75E-02$	$0.00E+00$	$7.44E+01$	$0.00E+00$	$3.61E+00$	$8.42E+01$	$4.63E+01$	$0.00E+00$
F_{13}	$1.68E+02$	$1.10E+02$	$5.75E+06$	$9.45E+01$	$1.49E+02$	$2.53E+02$	$3.16E+09$	$7.07E+01$
F_{14}	$9.76E-01$	$4.11E-16$	$4.29E+02$	$1.20E+01$	$4.75E+00$	$3.98E+02$	$4.09E+03$	$0.00E+00$
F_{15}	$0.00E+00$	$0.00E+00$	$2.14E+01$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$5.38E-03$	$0.00E+00$
F_{16}	$6.02E-02$	$0.00E+00$	$1.60E+02$	$0.00E+00$	$3.70E+00$	$1.97E+02$	$9.49E+00$	$0.00E+00$
F_{17}	$7.55E+01$	$2.39E+01$	$1.75E+05$	$8.39E-02$	$2.23E+01$	$6.07E+02$	$2.36E+05$	$9.31E+01$
F_{18}	$4.29E-04$	$2.04E-13$	$2.12E+02$	$8.93E-11$	$2.37E+00$	$2.34E+02$	$1.69E+01$	$0.00E+00$
F_{19}	$0.00E+00$	$0.00E+00$	$2.06E+03$	$0.00E+00$	$4.19E-02$	$0.00E+00$	$1.00E-01$	$0.00E+00$

TABLE A.22: Average error on 500D functions.

	MA-SSW-Chains	jDElscop	CHC	mDE-bES	DE	MTS-LS1	SaDE	FDA
F_1	$0.00E+00$	$0.00E+00$	$2.84E-12$	$3.92E-13$	$2.46E+01$	$5.77E-12$	$1.34E+01$	$0.00E+00$
F_2	$7.86E+01$	$3.06E+01$	$1.29E+02$	$4.56E+01$	$1.44E+02$	$5.34E-06$	$9.23E+01$	$4.30E-04$
F_3	$6.07E+02$	$4.06E+02$	$1.14E+06$	$4.16E+02$	$1.12E+05$	$2.20E+02$	$2.62E+04$	$5.82E+02$
F_4	$1.78E+02$	$1.59E-01$	$1.91E+03$	$1.91E-11$	$1.63E+01$	$5.62E-12$	$1.31E+00$	$0.00E+00$
F_5	$0.00E+00$	$0.00E+00$	$6.98E-03$	$1.83E-13$	$4.73E-01$	$4.24E-03$	$7.48E-03$	$0.00E+00$
F_6	$2.63E-13$	$1.18E-12$	$5.16E+00$	$3.56E-14$	$1.06E+00$	$6.18E-12$	$4.63E-01$	$8.74E-13$
F_7	$4.69E-14$	$0.00E+00$	$1.27E-01$	$0.00E+00$	$0.00E+00$	$1.46E-12$	$0.00E+00$	$0.00E+00$
F_8	$1.32E+04$	$5.66E+03$	$7.22E+04$	$5.48E+02$	$6.70E+04$	$6.16E+03$	$3.21E+05$	$0.00E+00$
F_9	$2.53E+03$	$6.10E-08$	$3.00E+03$	$0.00E+00$	$1.12E-02$	$1.00E+03$	$3.00E-02$	$0.00E+00$
F_{10}	$2.80E-01$	$0.00E+00$	$1.86E+02$	$0.00E+00$	$2.93E-01$	$0.00E+00$	$8.41E-03$	$0.00E+00$
F_{11}	$4.21E+01$	$4.40E-08$	$1.81E+03$	$0.00E+00$	$2.43E-01$	$1.00E+03$	$2.22E-03$	$0.00E+00$
F_{12}	$2.55E+01$	$0.00E+00$	$4.48E+02$	$0.00E+00$	$1.16E+01$	$2.47E+02$	$4.61E+01$	$0.00E+00$
F_{13}	$4.00E+02$	$3.14E+02$	$3.22E+07$	$3.23E+02$	$4.02E+02$	$5.05E+02$	$2.97E+09$	$3.74E+02$
F_{14}	$5.65E+01$	$8.00E-02$	$1.46E+03$	$1.68E+01$	$1.16E+01$	$1.10E+03$	$3.91E+03$	$0.00E+00$
F_{15}	$5.53E+00$	$0.00E+00$	$6.01E+01$	$0.00E+00$	$4.19E-02$	$1.08E-12$	$2.84E-03$	$0.00E+00$
F_{16}	$1.08E-01$	$0.00E+00$	$9.55E+02$	$0.00E+00$	$1.32E+01$	$4.99E+02$	$5.82E+00$	$0.00E+00$
F_{17}	$1.38E+02$	$7.65E+01$	$8.40E+05$	$6.65E+01$	$6.94E+01$	$7.98E+02$	$2.38E+05$	$3.96E+02$
F_{18}	$2.41E-03$	$1.11E-12$	$7.32E+02$	$0.00E+00$	$3.87E+00$	$5.95E+02$	$9.43E+00$	$0.00E+00$
F_{19}	$0.00E+00$	$0.00E+00$	$1.76E+03$	$0.00E+00$	$8.39E-02$	$0.00E+00$	$1.00E-01$	$0.00E+00$

TABLE A.23: Average error on 1000D functions.

	MA-SSW-Chains	jDElscop	CHC	mDE-bES	DE	MTS-LS1	SaDE	FDA
F_1	$0.00E+00$	$0.00E+00$	$1.36E-11$	$8.24E-13$	$3.71E+01$	$1.15E-11$	$3.49E+01$	$0.00E+00$
F_2	$1.39E+02$	$6.14E+01$	$1.44E+02$	$5.97E+01$	$1.63E+02$	$2.25E-02$	$1.43E+02$	$3.11E-01$
F_3	$1.22E+03$	$8.48E+02$	$8.75E+03$	$9.00E+02$	$1.59E+05$	$2.10E+02$	$1.62E+05$	$1.13E+03$
F_4	$1.58E+03$	$1.99E-01$	$4.76E+03$	$4.03E+01$	$3.47E+01$	$1.15E-11$	$3.21E+01$	$0.00E+00$
F_5	$5.92E-04$	$0.00E+00$	$7.02E-03$	$0.00E+00$	$7.36E-01$	$3.55E-03$	$6.33E-01$	$0.00E+00$
F_6	$1.46E-09$	$2.67E-12$	$1.38E+01$	$1.28E-12$	$8.70E-01$	$1.24E-11$	$4.28E-01$	$1.91E-12$
F_7	$6.23E-13$	$0.00E+00$	$3.52E-01$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
F_8	$7.49E+04$	$3.21E+04$	$3.11E+05$	$7.98E+03$	$3.15E+05$	$1.23E+05$	$3.08E+05$	$0.00E+00$
F_9	$5.99E+03$	$4.40E-03$	$6.11E+03$	$0.00E+00$	$6.26E-02$	$1.99E+03$	$3.00E-02$	$0.00E+00$
F_{10}	$2.09E-05$	$0.00E+00$	$3.83E+02$	$0.00E+00$	$1.67E-01$	$0.00E+00$	$1.47E-01$	$0.00E+00$
F_{11}	$5.27E+01$	$8.58E-04$	$4.82E+03$	$0.00E+00$	$4.42E-02$	$1.99E+03$	$4.56E-01$	$0.00E+00$
F_{12}	$9.48E-02$	$0.00E+00$	$1.05E+03$	$0.00E+00$	$2.58E+01$	$5.02E+02$	$3.43E+01$	$0.00E+00$
F_{13}	$1.02E+03$	$6.57E+02$	$6.66E+07$	$6.34E+02$	$8.24E+04$	$8.87E+02$	$3.27E+09$	$7.69E+02$
F_{14}	$7.33E+02$	$3.98E-02$	$3.62E+03$	$2.45E+01$	$2.39E+01$	$2.23E+03$	$3.71E+03$	$0.00E+00$
F_{15}	$1.16E-13$	$0.00E+00$	$8.37E+01$	$0.00E+00$	$2.11E-01$	$0.00E+00$	$1.11E-01$	$0.00E+00$
F_{16}	$2.19E+00$	$8.04E-01$	$2.32E+03$	$0.00E+00$	$1.83E+01$	$1.00E+03$	$2.37E+01$	$0.00E+00$
F_{17}	$3.26E+02$	$1.72E+02$	$2.04E+07$	$1.88E+02$	$1.76E+05$	$1.56E+03$	$1.62E+05$	$1.95E+02$
F_{18}	$2.58E+01$	$1.65E-01$	$1.72E+03$	$2.49E-01$	$7.55E+00$	$1.21E+03$	$3.54E+01$	$0.00E+00$
F_{19}	$0.00E+00$	$0.00E+00$	$4.20E+03$	$0.00E+00$	$2.51E-01$	$0.00E+00$	$9.32E+02$	$0.00E+00$

TABLE A.24: Average error on 50D functions for other methaeuristics

	MOS-SOCO2011	MOS-CEC2013	MOS-CEC2012	IACO$_{\mathbb{R}}$-Hybrid	2S-Ensemble	FDA
F_1	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
F_2	$5.88E-01$	$1.10E+02$	$1.03E+02$	$0.00E+00$	$4.31E+01$	$2.71E-12$
F_3	$7.09E+01$	$7.39E+00$	$9.38E+02$	$0.00E+00$	$1.34E+03$	$9.32E+01$
F_4	$0.00E+00$	$0.00E+00$	$1.90E+02$	$0.00E+00$	$8.58E-01$	$0.00E+00$
F_5	$0.00E+00$	$0.00E+00$	$1.18E-03$	$0.00E+00$	$3.00E-03$	$0.00E+00$
F_6	$0.00E+00$	$0.00E+00$	$1.03E+00$	$0.00E+00$	$0.00E+00$	$6.75E-14$
F_7	$0.00E+00$	$2.56E-12$	$1.03E-13$	$0.00E+00$	$Inf.$	$0.00E+00$
F_8	$1.66E+05$	$5.98E+03$	$1.09E+03$	$0.00E+00$	$1.93E+05$	$0.00E+00$
F_9	$0.00E+00$	$2.51E+03$	$5.95E+03$	$0.00E+00$	$2.68E+00$	$0.00E+00$
F_{10}	$0.00E+00$	$1.58E+00$	$1.79E+02$	$0.00E+00$	$0.00E+00$	$0.00E+00$
F_{11}	$0.00E+00$	$2.54E+03$	$5.88E+03$	$0.00E+00$	$3.23E+00$	$0.00E+00$
F_{12}	$0.00E+00$	$9.99E+02$	$1.12E+03$	$0.00E+00$	$0.00E+00$	$0.00E+00$
F_{13}	$1.69E+02$	$1.23E+03$	$2.03E+03$	$8.77E-01$	$1.25E+03$	$5.50E+01$
F_{14}	$0.00E+00$	$3.37E+03$	$4.32E+03$	$2.90E-02$	$4.40E-02$	$0.00E+00$
F_{15}	$0.00E+00$	$1.93E-12$	$2.04E+01$	$0.00E+00$	$Inf.$	$0.00E+00$
F_{16}	$0.00E+00$	$8.02E+03$	$2.33E+03$	$1.12E-03$	$0.00E+00$	$0.00E+00$
F_{17}	$6.71E+01$	$3.55E+11$	$3.71E+03$	$1.84E-06$	$3.39E+01$	$6.09E-04$
F_{18}	$0.00E+00$	$2.03E+03$	$2.29E+03$	$9.20E-01$	$5.51E-01$	$0.00E+00$
F_{19}	$0.00E+00$	$2.05E+03$	$5.25E+01$	$0.00E+00$	$7.99E-17$	$0.00E+00$

TABLE A.25: Number of times global optimum is reached for other metaheuristics. Values in parenthesis represent the algorithm's rank relative to the others among all dimensions

Dimensions	$D = 50$	Rank
MOS-SOCO2011	14	(1)
MOS-CEC2013	4	(5)
MOS-CEC2012	1	(6)
IACO$_{\mathbb{R}}$-Hybrid	14	(1)
2S-Ensemble	5	(4)
FDA	14	(1)

Appendix B

Tables of chapter 3 - PFDA

This appendix contains all tables regarding the Chapter 3

Table B.1: Results error of the 19 functions of SOCO 2011 for FDA and PFDA.

Function	Original FDA	NB Thread 4	NB Thread 8	NB Thread 16	NB Thread 32	NB Thread 64
$F1$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F2$	$3.11E-01$	$3.67E+01$	$5.43E+01$	$6.43E+01$	$7.19E+01$	$8.51E+01$
$F3$	$1.13E+03$	$1.41E+03$	$2.41E+03$	$3.24E+03$	$4.46E+03$	$4.63E+03$
$F4$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F5$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F6$	$1.91E-12$	$1.92E-12$	$1.92E-12$	$1.89E-12$	$1.92E-12$	$1.89E-12$
$F7$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F8$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F9$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F10$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F11$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F12$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.45E-14$
$F13$	$7.69E+02$	$9.78E+02$	$1.05E+03$	$1.08E+03$	$1.18E+03$	$1.55E+03$
$F14$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.45E-07$
$F15$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$
$F16$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$1.26E-07$
$F17$	$1.95E+02$	$3.76E+02$	$3.92E+02$	$4.30E+02$	$4.52E+02$	$6.88E+02$
$F18$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$5.89E-08$
$F19$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+00$	$0.00E+0$

Table B.2: Computation time required to solve the 19 functions of the benchmark used. Times are in seconds

Time in (s)	Original Extanded	MA-FDA-S1; l=3
N = 2	60	30
N = 5	160	30
N = 25	740	31
N = 50	1520	31

TABLE B.3: Experimental results obtained by the Original FDA, the extended FDA and MA-FDA-S1 for each level l from 1 to 5 and N = 2

	N = 2	Original	Original Extanded	MA-FDA-S1; l=1	MA-FDA-S1; l=2	MA-FDA-S1; l=3	MA-FDA-S1; l=4
50	Shifted Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem	2.71E-12	2.71E-12	2.71E-12	2.71E-12	2.71E-12	2.71E-12
50	Shifted Rosenbrock	9.32E+01	1.19E-02	9.32E+01	9.32E+01	8.47E+01	9.32E+01
50	Shifted Rastrigin	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Griewank	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Ackley	6.75E-14	6.39E-14	6.75E-14	6.75E-14	6.75E-14	6.75E-14
50	Schwefel Problem_2_22	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem_1_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	ExtendedF10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Bohachevsky	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schaffer	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 025	5.50E+01	5.43E+01	5.50E+01	5.50E+01	5.50E+01	5.50E+01
50	compF9 F4 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 075	6.09E-04	2.10E-06	5.94E-04	5.69E-04	6.09E-04	6.09E-04
50	compF9 F4 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE B.4: Experimental results obtained by the Original FDA, the extended FDA and MA-FDA-S1 for each level l from 1 to 5 and N = 5

	N = 5	Original	Original Extanded	MA-FDA-S1; l=1	MA-FDA-S1; l=2	MA-FDA-S1; l=3	MA-FDA-S1; l=4
50	Shifted Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem	2.71E-12	2.71E-12	2.71E-12	2.71E-12	2.71E-12	2.71E-12
50	Shifted Rosenbrock	9.32E+01	4.03E-10	9.32E+01	8.94E+01	8.41E+01	8.93E+01
50	Shifted Rastrigin	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Griewank	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Ackley	6.75E-14	6.39E-14	6.75E-14	6.75E-14	6.75E-14	6.75E-14
50	Schwefel Problem_2_22	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem_1_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	ExtendedF10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Bohachevsky	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schaffer	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 025	5.50E+01	5.22E+01	5.50E+01	5.50E+01	5.50E+01	5.50E+01
50	compF9 F4 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 075	6.09E-04	4.42E-14	5.94E-04	5.69E-04	5.76E-04	6.09E-04
50	compF9 F4 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE B.5: Experimental results obtained by the Original FDA, the extended FDA and MA-FDA-S1 for each level l from 1 to 5 and N = 25

	N = 25	Original	Original Extanded	MA-FDA-S1; l=1	MA-FDA-S1; l=2	MA-FDA-S1; l=3	MA-FDA-S1; l=4
50	Shifted Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem	2.71E-12	1.09E-12	2.71E-12	2.71E-12	2.71E-12	2.71E-12
50	Shifted Rosenbrock	9.32E+01	0.00E+00	3.94E+01	3.93E+01	3.92E+01	3.93E+01
50	Shifted Rastrigin	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Griewank	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Ackley	6.75E-14	6.39E-14	6.75E-14	6.75E-14	6.75E-14	6.75E-14
50	Schwefel Problem_2_22	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem_1_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	ExtendedF10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Bohachevsky	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schaffer	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 025	5.50E+01	3.83E+01	3.80E+00	2.39E+01	1.59E+00	2.53E+01
50	compF9 F4 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 075	6.09E-04	1.85E-23	5.70E-04	5.69E-04	3.01E-05	6.09E-04
50	compF9 F4 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE B.6: Experimental results obtained by the Original FDA, the extended FDA and MA-FDA-S1 for each level l from 1 to 5 and N = 50

N = 50		Original	Original Extanded	MA-FDA-S1; l=1	MA-FDA-S1; l=2	MA-FDA-S1; l=3	MA-FDA-S1; l=4
50	Shifted Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem	2.71E-12	1.09E-12	1.09E-12	1.09E-12	1.09E-12	1.09E-12
50	Shifted Rosenbrock	9.32E+01	0.00E+00	3.91E+01	3.93E+01	2.54E-01	1.53E-01
50	Shifted Rastrigin	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Griewank	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Ackley	6.75E-14	6.39E-14	6.39E-14	6.39E-14	6.39E-14	6.39E-14
50	Schwefel Problem_2_22	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem_1_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	ExtendedF10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Bohachevsky	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schaffer	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 025	5.50E+01	2.12E+01	3.80E+00	2.39E+01	1.59E+00	2.53E+01
50	compF9 F4 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 075	6.09E-04	0.00E+00	5.70E-04	5.28E-04	3.26E-08	6.09E-04
50	compF9 F4 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE B.7: MA-FDA-S1: Ranking using Friedman Rank sum of all variations at level $l = 1$, $l = 2$, $l = 3$, $l = 4$ and $D = 50$. (Values in parenthesis represent the algorithm's rank relative to the others)

	Original	Original Extanded	MA-FDA-S1; l=1	MA-FDA-S1; l=2	MA-FDA-S1; l=3	MA-FDA-S1; l=4
N = 2	1.5 (6)	1 (1)	1.47 (5)	1.37 (3)	1.32 (2)	1.42 (4)
N = 5	1.74 (6)	1 (1)	1.53 (4)	1.47 (3)	1.26 (2)	1.53 (4)
N = 25	1.84 (6)	1.21 (1)	1.53 (4)	1.47 (3)	1.21 (1)	1.58 (5)
N = 50	2.26 (6)	1.11 (1)	1.37 (3)	1.47 (4)	1.16 (2)	1.47 (4)

TABLE B.8: Experimental results obtained by the Original FDA and MA-FDA-S2 for each level l from 1 to 5 and N = 2

		Original	MA-FDA-S2; l=1	MA-FDA-S2; l=2	MA-FDA-S2; l=3	MA-FDA-S2; l=4
50	Shifted Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem	2.71E-12	1.42E-11	2.71E-12	2.71E-12	2.71E-12
50	Shifted Rosenbrock	9.32E+01	9.84E+01	9.74E+01	8.93E+01	9.79E+01
50	Shifted Rastrigin	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Griewank	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Ackley	6.75E-14	6.75E-14	6.75E-14	6.75E-14	6.75E-14
50	Schwefel Problem_2_22	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem_1_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	ExtendedF10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Bohachevsky	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schaffer	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 025	5.50E+01	5.50E+01	5.50E+01	5.50E+01	5.50E+01
50	compF9 F4 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 075	6.09E-04	6.12E-04	5.86E-04	6.45E-04	3.99E+00
50	compF9 F4 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE B.9: Experimental results obtained by the Original FDA and MA-FDA-S2 for each level l from 1 to 5 and N = 5

		Original	MA-FDA-S2; l=1	MA-FDA-S2; l=2	MA-FDA-S2; l=3	MA-FDA-S2; l=4
50	Shifted Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem	2.71E-12	2.71E-12	2.71E-12	2.71E-12	2.71E-12
50	Shifted Rosenbrock	9.32E+01	1.38E+02	1.32E+02	1.22E+02	1.31E+02
50	Shifted Rastrigin	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Griewank	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Ackley	6.75E-14	6.75E-14	6.75E-14	6.75E-14	6.75E-14
50	Schwefel Problem_2_22	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem_1_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	ExtendedF10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Bohachevsky	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schaffer	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 025	5.50E+01	5.50E+01	5.51E+01	5.50E+01	5.50E+01
50	compF9 F4 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 075	6.09E-04	7.91E-04	6.61E-04	8.57E-04	3.99E+00
50	compF9 F4 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE B.10: Experimental results obtained by the Original FDA and MA-FDA-S2 for each level l from 1 to 5 and N = 25

		Original	MA-FDA-S2; l=1	MA-FDA-S2; l=2	MA-FDA-S2; l=3	MA-FDA-S2; l=4
50	Shifted Sphere	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem	2.71E-12	2.71E-12	2.71E-12	2.71E-12	2.71E-12
50	Shifted Rosenbrock	9.32E+01	2.62E+02	2.52E+02	2.57E+02	2.66E+02
50	Shifted Rastrigin	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Griewank	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Shifted Ackley	6.75E-14	6.75E-14	6.75E-14	6.75E-14	7.46E-14
50	Schwefel Problem_2_22	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schwefel Problem_1_2	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	ExtendedF10	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Bohachevsky	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	Schaffer	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 025	5.50E+01	5.51E+01	5.51E+01	5.51E+01	5.51E+01
50	compF9 F4 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 025	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F1 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF9 F3 075	6.09E-04	9.76E+00	1.24E-02	1.32E-02	4.00E+00
50	compF9 F4 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
50	compF10 F7 075	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00

TABLE B.11: MA-FDA-S2: Ranking using Friedman Rank sum of all variations at level $l = 1$, $l = 2$, $l = 3$, $l = 4$ and $D = 50$.
(Values in parenthesis represent the algorithm's rank relative to the others

	Original	MA-FDA-S2; l = 1	MA-FDA-S2; l = 2	MA-FDA-S2; l = 3	MA-FDA-S2; l = 4
N = 2	1.21 (2)	1.74 (5)	1.26 (3)	1.16 (1)	1.42 (4)
N = 5	1.11 (1)	1.47 (5)	1.42 (4)	1.21 (2)	1.37 (3)
N = 25	1 (1)	1.47 (4)	1.26 (2)	1.26 (2)	1.79 (5)

Appendix C

Tables of chapter 4 - Mo-FDA

This appendix contains all tables regarding the Chapter 4

TABLE C.1: An example of computation time required to solve a function with Mo-FDA-S with different number of physical nodes N for 100 instances of FDA, hence 100 points in the Pareto Front. Times are in seconds

Number of Nodes	Time (in seconds)
Sequential Version	9.5
N = 2	8
N = 10	2
N = 25	0.8
N = 50	0.6

TABLE C.2: Definitions of the functions used from the ZDT Benchmark

ZDT1

$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$
$F_1(x) = x_1$
$F_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)}\right]$
$Subject: \ to: \ x \in [0,1]\,.$

ZDT2

$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$
$F_1(x) = x_1$
$F_2(x) = g(x)\left[1 - (x_1/g(x))^2\right]$
$Subject: \ to: \ x \in [0,1]\,.$

ZDT3

$g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$
$F_1(x) = x_1$
$F_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)} - x_1/g(x)\sin(10\pi x_1)\right]$
$Subject: \ to: \ x \in [0,1]\,.$

ZDT4

$g(x) = 91 + \sum_{i=2}^{n}\left[x_i^2 - 10\cos(4\pi x_i)\right]$
$F_1(x) = x_1$
$F_2(x) = g(x)\left[1 - \sqrt{x_1/g(x)}\right]$
$Subject: \ to: \ x_1 \in [0,1],$
$x_i \in [-5,5]\, i = 2, \cdots, 10.$

ZDT6

$g(x) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25}$
$F_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi\ x_1)$
$F_2(x) = g(x)\left[1 - (f_1(x)/g(x))^2\right]\ x \in [0,1]\,.$

TABLE C.3: Definitions of the functions used from the FTLZ Benchmark

DTLZ1

$$f_1(\vec{x}) = \frac{1}{2}x_1(1+g(\vec{x}))$$
$$f_2(\vec{x}) = \frac{1}{2}(1-x_1)(1+g(\vec{x}))$$
$$g(\vec{x}) = 100\left[|\vec{x}| + \sum_{x_i\in\vec{x}}(x_1-0.5)^2 - \cos(20\cdot\pi(x_i-0.5))\right]$$
$$0 \le x_i \le 1, i = 1,\dots,n$$

DTLZ2

$$f_1(\vec{x}) = (1+g(\vec{x}))\cos\left(x_1\frac{\pi}{2}\right)$$
$$f_2(\vec{x}) = (1+g(\vec{x}))\sin\left(x_1\frac{\pi}{2}\right)$$
$$g(\vec{x}) = \sum_{x_i\in\vec{x}}(x_i-0.5)^2$$
$$0 \le x_i \le 1, i = 1,\dots,n$$

DTLZ3

$$f_1(\vec{x}) = (1+g(\vec{x}))\cos\left(x_1\frac{\pi}{2}\right)$$
$$f_2(\vec{x}) = (1+g(\vec{x}))\sin\left(x_1\frac{\pi}{2}\right)$$
$$g(\vec{x}) = 100\cdot\left[|\vec{x}| + \sum_{x_i\in\vec{x}}(x_i-0.5)^2 - \cos(20\pi(x_i-0.5))\right]$$
$$0 \le x_i \le 1, i = 1,\dots,n$$

DLTZ4

$$f_1(\vec{x}) = (1+g(\vec{x}))\cos\left(x_1^\alpha\frac{\pi}{2}\right)$$
$$f_2(\vec{x}) = (1+g(\vec{x}))\sin\left(x_1^\alpha\frac{\pi}{2}\right)$$
$$g(\vec{x}) = \sum_{x_i\in\vec{x}}(x_i-0.5)^2$$
$$\alpha = 100$$
$$0 \le x_i \le 1, i = 1,\dots,n$$

TABLE C.4: Hypervolumes of the two studied scalarization methods on 4 different dimensions

	Weighted Sum	Tcheybycheff
D = 2	0.195509455	0.33616884
D = 5	0.042554689	0.31657224
D = 10	0.0425543400451	0.29477198
D = 30	0.136252213	0.31514439

TABLE C.5: Hypervolumes, ranks and computing time for the different studied Case from 1 to 9.

	Hypervolume	Rank	Computing Time (in s)
Case 1	0.308078317	8	0.58
Case 2	0.321532937	6	1.24
Case 3	0.326594895	5	1.06
Case 4	0.321305318	7	1.81
Case 5	0.337433922	1	4.95
Case 6	0.332881847	2	1.63
Case 7	0.329591937	3	13.89
Case 8	0.329558587	4	1.82

TABLE C.6: Hypervolumes, ranks and computing time for the different studied Cases from 9 to 14.

	Hypervolume	Computing Time (in s)
Case 9	0.285520947	0.84
Case 10	0.31573835	9.47
Case 11	0.208325898	1.34
Case 12	0.3289566	22.68
Case 13	0.155887101	6.46
Case 14	0.326810135011	15.13

TABLE C.7: Results for the different metrics for the function ZDT1. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	6.62E-01	**6.63E-01**	6.60E-01	6.24E-01	6.61E-01	6.58E-01	6.60E-01
GD	4.28E-05	**1.86E-05**	1.99E-04	2.20E-04	9.73E-05	2.24E-04	2.18E-03
IGD	1.98E-04	**1.19E-04**	1.86E-04	1.21E-03	1.60E-04	2.72E-04	2.05E-04
S	**2.75E-01**	1.15E+00	3.79E-01	4.23E-01	2.82E-01	8.77E-01	7.28E-01

TABLE C.8: Results for the different metrics for the function ZDT2. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	3.29E-01	**3.30E-01**	3.27E-01	2.95E-01	3.28E-01	3.24E-01	3.27E-01
GD	4.87E-05	**1.91E-05**	1.17E-04	1.11E-04	4.50E-05	5.77E-05	1.87E-03
IGD	1.40E-04	**1.39E-04**	1.95E-04	1.14E-03	1.41E-04	2.90E-04	4.62E-04
S	1.39E-01	1.10E+00	3.87E-01	3.94E-01	**1.35E-01**	2.61E-01	7.56E-01

TABLE C.9: Results for the different metrics for the function ZDT3. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	7.94E-01	**7.97E-01**	7.96E-01	7.77E-01	7.94E-01	7.92E-01	7.95E-01
GD	2.72E-03	**9.36E-05**	1.94E-04	6.43E-04	1.54E-04	1.73E-04	7.98E-03
IGD	9.16E-03	2.78E-04	**2.11E-04**	1.94E-03	4.70E-04	6.76E-04	4.77E-04
S	8.61E-01	1.11E+00	**5.57E-01**	6.35E-01	8.74E-01	1.13E+00	1.21E+00

TABLE C.10: Results for the different metrics for the function ZDT4. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	7.94E-01	**7.97E-01**	7.96E-01	7.77E-01	7.94E-01	7.92E-01	7.95E-01
GD	2.72E-03	**9.36E-05**	1.94E-04	6.43E-04	1.54E-04	1.73E-04	7.98E-03
IGD	9.16E-03	2.78E-04	**2.11E-04**	1.94E-03	4.70E-04	6.76E-04	4.77E-04
S	8.61E-01	1.11E+00	**5.57E-01**	6.35E-01	8.74E-01	1.13E+00	1.21E+00

TABLE C.11: Results for the different metrics for the function ZDT6. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	3.19E-01	3.18E-01	3.20E-01	2.98E-01	**3.22E-01**	3.19E-01	3.20E-01
GD	3.46E-05	**1.46E-05**	5.72E-05	1.45E-04	8.01E-05	6.00E-05	3.56E-05
IGD	3.24E-04	3.72E-04	1.84E-04	9.36E-04	**1.17E-04**	2.24E-04	1.95E-04
S	3.01E-01	1.46E+00	6.44E-01	4.80E-01	**1.34E-01**	9.49E-01	4.82E-01

TABLE C.12: Results for the different metrics for the function DTLZ1 with 2 Objectives function. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	8.71E-01	8.71E-01	**8.72E-01**	0.00E+00	8.72E-01	8.71E-01	0.00E+00
GD	2.06E-05	**7.41E-06**	1.97E-04	5.61E-01	2.76E-04	2.08E-04	9.27E+00
IGD	6.57E-05	**5.28E-05**	1.13E-04	5.91E-02	1.08E-04	1.38E-04	2.52E+00
S	**1.07E-02**	1.11E+00	4.49E-01	8.32E-01	1.57E-02	6.77E-02	8.96E-01

TABLE C.13: Results for the different metrics for the function DTLZ2 with 2 Objectives function. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2.10E-01	**2.13E-01**	2.09E-01	1.81E-01	2.10E-01	2.06E-01	2.08E-01
GD	3.46E-04	**4.23E-05**	2.64E-04	4.95E-04	3.50E-04	2.59E-04	9.36E-05
IGD	1.72E-04	**8.53E-05**	1.92E-04	1.42E-03	1.73E-04	3.30E-04	3.33E-04
S	1.84E-01	1.11E+00	3.87E-01	4.08E-01	**1.81E-01**	2.74E-01	6.88E-01

TABLE C.14: Results for the different metrics for the function DTLZ3 with 2 Objectives function. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2.10E-01	**2.13E-01**	2.05E-01	0.00E+00	2.01E-01	2.02E-01	0.00E+00
GD	2.44E-04	**3.18E-05**	3.55E-04	1.06E+00	6.70E-04	4.07E-04	1.25E+01
IGD	**1.55E-04**	1.93E-04	2.24E-04	1.33E-01	2.51E-04	3.39E-04	3.95E+00
S	**1.84E-01**	1.11E+00	4.08E-01	8.81E-01	1.93E-01	2.16E-01	8.71E-01

TABLE C.15: Results for the different metrics for the function DTLZ4 with 2 Objectives function. Values in bold represent the best value for each metric

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	**2.10E-01**	1.70E-01	2.09E-01	9.14E-03	2.10E-01	2.08E-01	2.06E-01
GD	2.44E-04	2.26E-04	1.83E-04	**1.45E-05**	2.35E-04	1.11E-04	1.71E-04
IGD	**1.56E-04**	2.51E-03	2.05E-04	3.01E-02	1.56E-04	7.62E-04	3.16E-04
S	**1.83E-01**	1.56E+00	4.12E-01	9.69E-01	1.84E-01	6.77E-01	3.38E-01

TABLE C.16: Ranks for each metric and each algorithm the function ZDT1.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2	1	4	7	3	6	5
GD	2	1	4	5	3	6	7
IGD	4	1	3	7	2	6	5
S	1	7	3	4	2	6	5

TABLE C.17: Ranks for each metric and each algorithm the function ZDT2.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2	1	4	7	3	6	5
GD	3	1	6	5	2	4	7
IGD	2	1	4	7	3	5	6
S	2	7	4	5	1	3	6

TABLE C.18: Ranks for each metric and each algorithm the function ZDT3.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	4	1	2	7	5	6	3
GD	6	1	4	5	2	3	7
IGD	7	2	1	6	3	5	4
S	3	5	1	2	4	6	7

Table C.19: Ranks for each metric and each algorithm the function ZDT4.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2	1	3	6	5	4	7
GD	2	1	3	6	5	4	7
IGD	4	1	2	6	3	5	7
S	1	7	3	4	2	5	6

Table C.20: Ranks for each metric and each algorithm the function ZDT6.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	4	6	2	7	1	5	3
GD	2	1	4	7	6	5	3
IGD	5	6	2	7	1	4	3
S	2	7	5	3	1	6	4

Table C.21: Ranks for each metric and each algorithm the function DTLZ1 for 2 Objectives.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	5	4	1	6	2	3	6
GD	2	1	3	6	5	4	7
IGD	2	1	4	6	3	5	7
S	1	7	4	5	2	3	6

Table C.22: Ranks for each metric and each algorithm the function DTLZ2 for 2 Objectives.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2	1	4	7	3	6	5
GD	5	1	4	7	6	3	2
IGD	2	1	4	7	3	5	6
S	2	7	4	5	1	3	6

Table C.23: Ranks for each metric and each algorithm the function DTLZ3 for 2 Objectives.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2	1	3	6	5	4	6
GD	2	1	3	6	5	4	7
IGD	1	2	3	6	4	5	7
S	1	7	4	6	2	3	5

Table C.24: Ranks for each metric and each algorithm the function DTLZ4 for 2 Objectives.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	1	6	3	7	2	4	5
GD	7	5	4	1	6	2	3
IGD	1	6	3	7	2	5	4
S	1	7	4	6	2	5	3

TABLE C.25: Average ranks for each metric and each algorithm over the 9 functions used using the Friendman Rank Rum.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2.67	2.44	2.89	6.67	3.22	4.89	5.00
GD	3.44	1.44	3.89	5.33	4.44	3.89	5.56
IGD	3.11	2.33	2.89	6.56	2.67	5.00	5.44
S	1.56	6.78	3.56	4.44	1.89	4.44	5.33

TABLE C.26: Final ranks based on the Friendman Rank Sum values for the 9 used functions.

	Mo-FDA-S	Mo-FDA-D	NSGA2	NSGA3	MOEAD	GWASGFA	CDG
HV	2	1	3	7	4	5	6
GD	2	1	3	6	5	3	7
IGD	4	1	3	7	2	5	6
S	1	6	3	4	2	4	6
Final Average Rank:	2.25	2.25	3	6	3.25	4.25	6.25

TABLE C.27: Results for the different metrics for the function DTLZ1 with 3 Objectives. Values in bold represent the best value for each metric

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	9.46E-01	9.47E-01	**9.72E-01**	9.67E-01	5.75E-01
GD	**6.16E-05**	4.55E-03	7.18E-04	6.47E-04	2.66E+01
IGD	2.48E-04	5.94E-04	**2.39E-04**	4.00E-04	2.49E-03
S	1.20E+00	8.44E-01	**5.94E-01**	7.86E-01	6.48E-01

TABLE C.28: Results for the different metrics for the function DTLZ2 with 3 Objectives. Values in bold represent the best value for each metric

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	3.97E-01	3.74E-01	**4.14E-01**	3.78E-01	3.77E-01
GD	**1.54E-04**	1.39E-03	7.62E-04	1.04E-03	3.79E-02
IGD	6.52E-04	7.77E-04	**5.95E-04**	1.04E-03	5.95E-04
S	1.28E+00	6.94E-01	**5.95E-01**	7.47E-01	6.33E-01

TABLE C.29: Results for the different metrics for the function DTLZ3 with 3 Objectives. Values in bold represent the best value for each metric.

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	**3.97E-01**	3.59E-01	3.92E-01	3.54E-01	0.00E+00
GD	**2.93E-04**	1.93E-03	1.84E-03	1.66E-03	6.71E+01
IGD	1.07E-03	1.30E-03	**9.94E-04**	1.71E-03	3.20E+00
S	1.29E+00	7.31E-01	5.89E-01	8.29E-01	**4.92E-01**

Table C.30: Results for the different metrics for the function DTLZ4 with 3 Objectives. Values in bold represent the best value for each metric

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	2.08E-01	3.60E-01	2.56E-01	**3.61E-01**	3.20E-01
GD	**2.04E-04**	4.82E-03	4.28E-03	8.14E-03	5.10E-02
IGD	2.91E-03	**1.66E-03**	4.34E-03	1.93E-03	1.92E-03
S	1.71E+00	6.94E-01	8.12E-01	7.65E-01	**6.42E-01**

Table C.31: Ranks for each metric and each algorithm the function DTLZ1 for 3 Objectives.

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	4	3	1	2	5
GD	1	4	3	2	5
IGD	2	4	1	3	5
S	5	4	1	3	2

Table C.32: Ranks for each metric and each algorithm the function DTLZ2 for 3 Objectives.

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	2	5	1	3	4
GD	1	4	2	3	5
IGD	3	4	1	5	2
S	5	3	1	4	2

Table C.33: Ranks for each metric and each algorithm the function DTLZ3 for 3 Objectives.

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	1	3	2	4	5
GD	1	4	3	2	5
IGD	2	3	1	4	5
S	5	3	2	4	1

Table C.34: Ranks for each metric and each algorithm the function DTLZ4 for 3 Objectives.

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	5	2	4	1	3
GD	1	3	2	4	5
IGD	4	1	5	3	2
S	5	2	4	3	1

Table C.35: Average ranks for each metric and each algorithm over the 4 DTLZ 3-Objective functions used using the Friendman Rank Rum.

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	3	3.25	2	2.5	4.25
GD	1	3.75	2.5	2.75	5
IGD	2.75	3	2	3.75	3.5
S	5	3	2	3.5	1.5

TABLE C.36: Final ranks based on the Friendman Rank Sum values over the 4 DTLZ 3-Objective functions.

	Mo-FDA-D	NSGA2	NSGA3	GWASGFA	CDG
HV	3	4	1	2	5
GD	1	4	2	3	5
IGD	2	3	1	5	4
S	5	3	2	4	1
Final Average Rank:	2.75	3.5	1.5	3.5	3.75

Appendix D

Results on CIFAR-100

CIFAR-100

As a final performance test, we decided to apply our best results found to the similar but significantly more complex benchmark, CIFAR-100. It is built similarly to CIFAR-10 with a training set of 50000 images and a test set of 10000 images. However, those images are classified into 100 different classes in opposition to the 10 classes of CIFAR-10. Classes are grouped into 20 superclasses. Each image has a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs). Table D.1 shows the superclasses and classes and Figure D.1 illustrates images with both their Superclass and class (the format shown is *Superclass-class*).

On this data set, using our best architecture and parameters, we reached a validation accuracy of 67.65% which is, proportionally to the size of the architecture, close to some state-of-the-art neural networks.

Table D.1: Labels for CIFAR-100 benchmark divided into Superclasses and Classes

Superclass	**Classes**
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

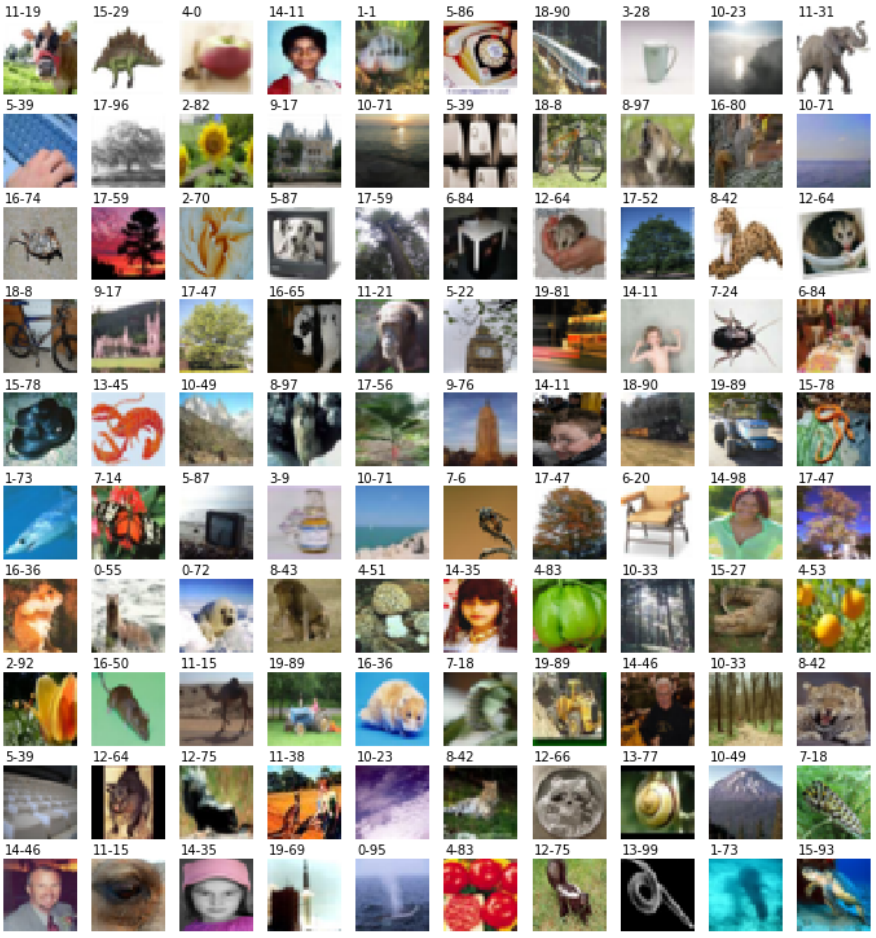


FIGURE D.1: Examples of images in CIFAR-100 with their Superclass-Class

Bibliography

[Abadi et al., 2016] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, & Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016. URL https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[A.Bastürk & E.Günay, 2009] A.Bastürk & E.Günay. Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm. *Expert System with Applications*, 36(2): 2645–2650, 2009.

[Aimin Zhou et al., 2006] Aimin Zhou, Yaochu Jin, Qingfu Zhang, B. Sendhoff, & E. Tsang. Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In *2006 IEEE International Conference on Evolutionary Computation*, pp. 892–899, July 2006. doi: 10.1109/CEC.2006.1688406.

[Akhmetova et al., 2017] Dana Akhmetova, Roman Iakymchuk, Orjan Ekeberg, & Erwin Laure. Performance study of multithreaded mpi and openmp tasking in a large scientific code. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 756–765, May 2017.

[Al-Dujaili et al., 2016a] Abdullah Al-Dujaili, S. Suresh, & N. Sundararajan. Mso: A framework for bound-constrained black-box global optimization algorithms. *J. of Global Optimization*, 66(4): 811–845, Dec 2016.

[Al-Dujaili et al., 2016b] Abdullah Al-Dujaili, Sundaram Suresh, & Narasimhan Sundararajan. MSO: a framework for bound-constrained black-box global optimization algorithms. *Journal of Global Optimization*, 66(4): 811–845, December 2016. doi: 10.1007/s10898-016-0441-5.

[Alba & Luque, 2006] Enrique Alba & Gabriel Luque. Evaluation of parallel metaheuristics. 2006.

[Alefeld & Mayer, 2000] Götz Alefeld & Günter Mayer. Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*, 121(1): 421 – 464, 2000. ISSN 0377-0427. doi: https://doi.org/10.1016/S0377-0427(00)00342-3.

[Ali et al., 2015] Mostafa Z Ali, Noor H Awad, & Ponnuthurai N Suganthan. Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization. *Applied Soft Computing*, 33: 304–327, 2015.

[Apruzzese et al., 2018] Giovanni Apruzzese, Michele Colajanni, Luca Ferretti, Alessandro Guido, & Mirco Marchetti. On the effectiveness of machine and deep learning for cyber security. In *2018 10th International Conference on Cyber Conflict (CyCon)*, pp. 371–390, May 2018. doi: 10.23919/CYCON.2018.8405026.

[Araya & Reyes, 2015] Ignacio Araya & Victor Reyes. Interval branch-and-bound algorithms for optimization and constraint satisfaction: a survey and prospects. *Journal of Global Optimization*, 65, 12 2015. doi: 10.1007/s10898-015-0390-4.

[Arnautovic et al., 2013] Maida Arnautovic, Maida Curic, Emina Dolamic, & Novica Nosovic. Parallelization of the ant colony optimization for the shortest path problem using openmp and cuda. In *2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1273–1277, May 2013.

[Ashlock & Schonfeld, 2007] Daniel Ashlock & Justin Schonfeld. A fractal representation for real optimization. In *2007 IEEE Congress on Evolutionary Computation*, pp. 87–94, sep 2007.

[Baker et al., 2017] Bowen Baker, Otkrist Gupta, Nikhil Naik, & Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017*, pp. 1–9, April 2017.

[Baocheng et al., 2012] Wan Baocheng, Wang Tiane, & Wang Zenghui. The implementation of parallel ant colony optimization algorithm based on matlab. In *2012 Third Global Congress on Intelligent Systems*, pp. 27–29, Nov 2012.

[Bertsekas, 1995] Dimitri Bertsekas. Nonlinear programming. *Athena Scientific*, 48, 01 1995. doi: 10.1057/palgrave.jors.2600425.

[Bertsimas & Tsitsiklis, 1998] Dimitris Bertsimas & John Tsitsiklis. *Introduction to Linear Optimization*. 01 1998.

[Blum et al., 2008] Christian Blum, Carlos Cotta, Antonio J. Fernández, José E. Gallardo, & Monaldo Mastrolilli. *Hybridizations of Metaheuristics With Branch & Bound Derivates*, pp. 85–116. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-78295-7. doi: 10.1007/978-3-540-78295-7_4. URL https://doi.org/10.1007/978-3-540-78295-7_4.

[Boyd & Vandenberghe, 2004] Stephen Boyd & Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

[Brest & Maučec, 2011] Janez Brest & Mirjam Sepesy Maučec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing*, 15(11): 2157–2174, nov 2011.

[Brunetti et al., 2018] Antonio Brunetti, Domenico Buongiorno, Gianpaolo Francesco Trotta, & Vitoantonio Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300: 17 – 33, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.01.092. URL http://www.sciencedirect.com/science/article/pii/S092523121830290X.

[Bäck et al., 2000] Thomas Bäck, T. Fogel, & D. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol, 2000.

[Cahon et al., 2004] S. Cahon, El-Ghazali Talbi, & Nouredine Melab. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3): 357–380, May 2004. ISSN 1572-9397. doi: 10.1023/B:HEUR.0000026900.92269.ec. URL https://doi.org/10.1023/B:HEUR.0000026900.92269.ec.

[Cai et al., 2018] Xinye Cai, Zhiwei Mei, Zhun Fan, & Qingfu Zhang. A constrained decomposition approach with grids for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 22(4): 564–577, Aug 2018.

[Camero et al., 2019] Andrés Camero, Jamal Toutouh, & Enrique Alba. A specialized evolutionary strategy using mean absolute error random sampling to design recurrent neural networks, 2019.

[Chang, 2006] Wei-Der Chang. Parameter identification of rosslerś chaotic system by an evolutionary algorithm. *Chaos, Solitons and Fractals*, 29(5): 1047–1053, 2006.

[Chelouah & Siarry, 1999] Rachid Chelouah & Patrick Siarry. *Enhanced Continuous Tabu Search: An Algorithm for Optimizing Multiminima Functions*, pp. 49–61. Springer US, Boston, MA, 1999. ISBN 978-1-4615-5775-3. doi: 10.1007/978-1-4615-5775-3_4. URL https://doi.org/10.1007/978-1-4615-5775-3_4.

[Chelouah & Siarry, 2000] Rachid Chelouah & Patrick Siarry. A continuous genetic algorithm designed for the global optimization of multimodal functions. *Journal of Heuristics*, 6(2): 191–213, Jun 2000. ISSN 1572-9397. doi: 10.1023/A:1009626110229. URL https://doi.org/10.1023/A:1009626110229.

[Clautiaux et al., 2004] François Clautiaux, Aziz Moukrim, Stéphane Nègre, & Jacques Carlier. Heuristic and metaheuristic methods for computing graph treewidth. *RAIRO - Operations Research*, 38(1): 13–26, 2004. doi: 10.1051/ro:2004011.

[Cung et al., 2001] Van-Dat Cung, Simone Martins, Caio Ribeiro, & Catherine Roucairol. Strategies for the parallel implementation of metaheuristics. 01 2001. doi: 10.1007/978-1-4615-1507-4_13.

[Darwin, 1859] Charles Darwin. *On the Origin of Species by Means of Natural Selection*. Murray, London, 1859. or the Preservation of Favored Races in the Struggle for Life.

[Darwish et al., 2019] Ashraf Darwish, Aboul Ella Hassanien, & Swagatam Das. A survey of swarm and evolutionary computing approaches for deep learning. *Artificial Intelligence Review*, Jun 2019.

[Das et al., 2016] Swagatam Das, Sankha Subhra Mullick, & P.N. Suganthan. Recent advances in differential evolution – an updated survey. *Swarm and Evolutionary Computation*, 27: 1 – 30, 2016. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2016.01.004.

[Deb et al., 2002a] Kalyan Deb, L. Thiele, Marco Laumanns, & Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC 2002)*, pp. 825–830. IEEE Press, 2002.

[Deb & Jain, 2014] Kalyanmoy Deb & Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4): 577–601, Aug 2014. ISSN 1089-778X. doi: 10.1109/TEVC.2013.2281535.

[Deb et al., 2002b] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, & T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2): 182–197, April 2002.

[Deb et al., 2007] Kalyanmoy Deb, Udaya Bhaskara Rao N., & S. Karthik. Dynamic multi-objective optimization and decision-making using modified nsga-ii: A case study on hydro-thermal power scheduling. In Shigeru Obayashi, Kalyanmoy Deb, Carlo Poloni, Tomoyuki Hiroyasu, & Tadahiko Murata, editors, *Evolutionary Multi-Criterion Optimization*, pp. 803–817, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-70928-2.

[Delisle et al., 2009] Pierre Delisle, Marc Gravel, & Michael Krajecki. Multi-colony parallel ant colony optimization on smp and multi-core computers. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pp. 318–323, Dec 2009.

[Demirhan et al., 1999] Melek Demirhan, Linet Özdamar, Levent Helvacğlu, & Şevket Ilker Birbil. Fractop: A geometric partitioning metaheuristic for global optimization. *J. of Global Optimization*, 14 (4): 415–436, June 1999.

[Demirhan & Özdamar, 2003] Melek Basak Demirhan & Linet Özdamar. *A Fuzzy Adaptive Partitioning Algorithm (FAPA) for Global Optimization*, pp. 37–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-36461-0. doi: 10.1007/978-3-540-36461-0_3. URL https://doi.org/10.1007/978-3-540-36461-0_3.

[Deng et al., 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, & Fei Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, June 2009. doi: 10.1109/CVPR.2009.5206848.

[Di Domenico et al., 2017] Daniel Di Domenico, Joao Lima, & Andrea Charao. Openmp with parallel loops or asynchronous tasks: a performance evaluation focusing the nqueens benchmark. *IEEE Latin America Transactions*, 15(9): 1793–1800, 2017.

[Domhan et al., 2015] Tobias Domhan, Jost Tobias Springenberg, & Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 3460–3468. AAAI Press, 2015. ISBN 978-1-57735-738-4. URL http://dl.acm.org/citation.cfm?id=2832581.2832731.

[Durillo & Nebro, 2011] Juan J. Durillo & Antonio J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10): 760 – 771, 2011. ISSN 0965-9978.

[Elsken et al., 2018] Thomas Elsken, Jan Hendrik Metzen, & Frank Hutter. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20: 55:1–55:21, 2018.

[Eshelman & Schaffer, 1992] Larry J Eshelman & J David Schaffer. Real-Coded Genetic Algorithms and Interval-Schemata. In L Darrell Whitley, editor, *FOGA*, pp. 187–202. Morgan Kaufmann, 1992. ISBN 1-55860-263-1.

[Esteva et al., 2019] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, & Jeff Dean. A guide to deep learning in healthcare. *Nature Medicine*, 25, 01 2019. doi: 10.1038/s41591-018-0316-z.

[Finkel & Kelley, 2006] Daniel Finkel & Carl Kelley. Additive scaling and the direct algorithm. *Journal of Global Optimization*, 36(4): 597–608, Dec 2006. ISSN 1573-2916. doi: 10.1007/s10898-006-9029-9. URL https://doi.org/10.1007/s10898-006-9029-9.

[Fogel et al., 1966] Lawrence Fogel, Alvin Owens, & Michael Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, 1966.

[Fong et al., 2018] Simon Fong, Suash Deb, & Xin-She Yang. *How Meta-heuristic Algorithms Contribute to Deep Learning in the Hype of Big Data Analytics*, pp. 3–25. 01 2018. ISBN 978-981-10-3372-8. doi: 10.1007/978-981-10-3373-5_1.

[Fonseca & Fleming, 1995] Carlos M. Fonseca & Peter J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1): 1–16, 1995.

[Gablonsky & Kelley, 2001] Joerg Gablonsky & Carl Kelley. A locally-biased form of the direct algorithm. *Journal of Global Optimization*, 21: 27–37, 01 2001. doi: 10.1023/A:1017930332101.

[Geilen & Basten, 2007] Marc Geilen & Twan Basten. A calculator for pareto points. In *2007 Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, April 2007.

[Glover, 1986] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13 (5): 533 – 549, 1986. ISSN 0305-0548. doi: https://doi.org/10.1016/0305-0548(86)90048-1. URL http://www.sciencedirect.com/science/article/pii/0305054886900481. Applications of Integer Programming.

[Glover, 1997] Fred Glover. *Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges*, pp. 1–75. Springer US, Boston, MA, 1997. ISBN 978-1-4615-4102-8. doi: 10.1007/978-1-4615-4102-8_1. URL https://doi.org/10.1007/978-1-4615-4102-8_1.

[Goldberg, 1989] David E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 1st edition, 1989.

[Gong et al., 2015] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, & Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34: 286 – 300, 2015. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2015.04.061. URL http://www.sciencedirect.com/science/article/pii/S1568494615002987.

[Gorges-Schleuter, 1989] Martina Gorges-Schleuter. Asparagos an asynchronous parallel genetic optimization strategy. In *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 422–427, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc. ISBN 1-55860-006-3.

[Hajji et al., 2004] Oussama Hajji, S. Brisset, & Pascal Brochet. A new tabu search method for optimization with continuous parameters. *IEEE Transactions on Magnetics*, 40(2): 1184–1187, March 2004. ISSN 0018-9464. doi: 10.1109/TMAG.2004.824909.

[Hansen & Ostermeier, 2001] Nikolaus Hansen & Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2): 159–195, June 2001. ISSN 1063-6560. doi: 10.1162/106365601750190398. URL http://dx.doi.org/10.1162/106365601750190398.

[Hansen et al., 1991] Pierre Hansen, Brigitte Jaumard, & Shi-Hui Lu. An analytical approach to global optimization. *Math. Program.*, 52(1-3): 227–254, May 1991. ISSN 0025-5610. doi: 10.1007/BF01582889. URL https://doi.org/10.1007/BF01582889.

[Hasançebi et al., 2010] Oguzhan Hasançebi, Serdar Carbas, & Mehmet Saka. Improving the performance of simulated annealing in large-scale structural optimization. *Structural and Multidisciplinary Optimization*, 41: 189–203, 03 2010. doi: 10.1007/s00158-009-0418-9.

[He et al., 2004] J He, M. Sosonkina, Clifford Shaffer, J Tyson, L.T. Watson, & Jason Zwolak. Hierarchical parallel scheme for global parameter estimation in systems biology. In *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, pp. 42–, April 2004.

[He et al., 2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision (ICCV 2015)*, 1502, 02 2015. doi: 10.1109/ICCV.2015.123.

[Hedar & Fouad, 2012] Abdel-Rahman Hedar & Ahmed Fouad. Tabu search with multi-level neighborhood structures for high dimensional problems. *Applied Intelligence*, 37, 09 2012. doi: 10.1007/s10489-011-0321-0.

[Helbig, 2016] Mardé Helbig. Updating the global best and archive solutions of the dynamic vector-evaluated pso algorithm using ϵ-dominance. In Nelishia Pillay, Andries P. Engelbrecht, Ajith Abraham, Mathys C. du Plessis, Václav Snášel, & Azah Kamilah Muda, editors, *Advances in Nature and Biologically Inspired Computing*, pp. 393–403, Cham, 2016. Springer International Publishing. ISBN 978-3-319-27400-3.

[Herrera et al., 2017] Juan F. R. Herrera, José M. G. Salmerón, Eligius M. T. Hendrix, Rafael Asenjo, & Leocadio G. Casado. On parallel branch and bound frameworks for global optimization. *Journal of Global Optimization*, Mar 2017.

[Hinton, 2012] Geoffrey Hinton. Lecture 6a - overview of mini-batch gradient descent, 2012.

[Holland, 1975] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.

[Hooke & Jeeves, 1961] Robert Hooke & T. A. Jeeves. "direct search" solution of numerical and statistical problems. *J. ACM*, 8(2): 212–229, April 1961. ISSN 0004-5411.

[Ioffe & Szegedy, 2015] Sergey Ioffe & Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 448–456. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045167.

[Ishibuchi et al., 2015] Hisao Ishibuchi, Hiroyuki Masuda, & Yusuke Nojima. A study on performance evaluation ability of a modified inverted generational distance indicator. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pp. 695–702, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3472-3. doi: 10.1145/2739480.2754792. URL http://doi.acm.org/10.1145/2739480.2754792.

[Jaszkiewicz, 2002] Andrzej Jaszkiewicz. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment. *IEEE Transactions on Evolutionary Computation*, 6(4): 402–412, Aug 2002. ISSN 1089-778X. doi: 10.1109/TEVC.2002.802873.

[Jiang et al., 2014] Siwei Jiang, Jie Zhang, & Yew Ong. Multiobjective optimization based on reputation. *Information Sciences*, 286: 125 – 146, 2014. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2014.07.020. URL http://www.sciencedirect.com/science/article/pii/S0020025514007208.

[Jones et al., 1993] D R Jones, C D Perttunan, & B E Stuckman. Lipshitzian optimization without the Lipshitz coefficient. *Journal of Optimization Theory Applications*, 79(1): 157–181, 1993.

[Kečo et al., 2016] Dino Kečo, Abdulhamit Subasi, & Jasmin Kevric. Cloud computing-based parallel genetic algorithm for gene selection in cancer classification. *Neural Computing and Applications*, 12 2016. doi: 10.1007/s00521-016-2780-z.

[Kingma & Ba, 2014] Diederik Kingma & Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[Kirkpatrick, 1984] Scott Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of statistical physics*, 34: 975–986, 1984.

[Kirkpatrick et al., 1983] Scott Kirkpatrick, Charles Daniel Gelatt, & Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220 4598: 671–80, 1983.

[Klein et al., 2016] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, & Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. 05 2016.

[Krizhevsky et al., 2012] Alex Krizhevsky, Ilya Sutskever, & Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

[Lakshmi et al., 2011] Dhana Lakshmi, Kannan Subramanian, K. Mahadevan, & Baskar Subramanian. Application of modified nsga-ii algorithm to combined economic and emission dispatch problem. *International Journal of Electrical Power & Energy Systems*, 33(4): 992 – 1002, 2011. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2011.01.014. URL http://www.sciencedirect.com/science/article/pii/S0142061511000421.

[Lamarck, 1830] Jean Baptiste Pierre Antoine de Monet de 1744-1829 Lamarck. *Philosophie zoologique, ou Exposition des considérations relatives à l'histoire naturelle des animaux ...* Nouvelle édition. Paris : J. B. Baillière; Londres : Même Maison; Bruxelles : Au Dépôt de la Librairie Médical Française, 1830., 1830. URL https://search.library.wisc.edu/catalog/999596463002121.

[Land & Doig, 1960] Ailsa Land & Alison Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3): 497–520, 1960. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1910129.

[Lang et al., 2007] Haitao Lang, Liren Liu, & Qingguo Yang. A novel method to design flexible uras. *Journal of Optics A: Pure and Applied Optics*, 9: 502, 04 2007. doi: 10.1088/1464-4258/9/5/014.

[LaTorre et al., 2011] Antonio LaTorre, Santiago Muelas, & José-Mar\'\ia Peña. A MOS-based dynamic memetic differential evolution algorithm for

continuous optimization: a scalability test. *Soft Computing*, 15 (11): 2187–2199, 2011.

[LaTorre et al., 2013] Antonio LaTorre, Santiago Muelas, & Jose-Maria Pena. Large scale global optimization: Experimental results with MOS-based hybrid algorithms. In *2013 IEEE Congress on Evolutionary Computation*, pp. 2742–2749, June 2013.

[LaTorre et al., 2014] Antonio LaTorre, Santiago Muelas, & José-María Peña. A comprehensive comparison of large scale global optimizers. *Information Sciences*, 316(0): 517—549, 2014.

[Lecun et al., 1998] Yann Lecun, Leon Bottou, Y. Bengio, & Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, Nov 1998. doi: 10.1109/5.726791.

[LeCun et al., 2015] Yann LeCun, Y Bengio, & Geoffrey Hinton. Deep learning. *Nature*, 521: 436–44, 05 2015. doi: 10.1038/nature14539.

[Lee et al., 2014] Loo Hay Lee, Ek Peng Chew, Yu Qian, Haobin Li, & Yue Liu. A study on multi-objective particle swarm optimization with weighted scalarizing functions. In *Proceedings of the Winter Simulation Conference 2014*, pp. 3718–3729, Dec 2014. doi: 10.1109/WSC.2014.7020200.

[Li & Zhang, 2009] Hui Li & Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on Evolutionary Computation*, 13(2): 284–302, April 2009. ISSN 1089-778X. doi: 10.1109/TEVC.2008.925798.

[Li & Talwalkar, 2019] Liam Li & Ameet Talwalkar. Random search and reproducibility for neural architecture search, 02 2019.

[Liao et al., 2011] Tianjun Liao, Marco A. Montes de Oca, Dogan Aydin, Thomas Stützle, & Marco Dorigo. An incremental ant colony algorithm with local search for continuous optimization. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pp. 125–132. ACM, 2011.

[Liu & He, 2012] Huibin Liu & Zhenfeng He. Parallel ant colony optimization algorithms for time series segmentation on a multi-core processor. In *2012 4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1, pp. 340–343, Aug 2012.

[Liu & Cheng, 2014] Qunfeng Liu & Wanyou Cheng. A modified direct algorithm with bilevel partition. *Journal of Global Optimization*, 60: 483–499, 11 2014. doi: 10.1007/s10898-013-0119-1.

[Liu et al., 2015] Qunfeng Liu, Jinping Zeng, & Gang Yang. Mrdirect: a multilevel robust direct algorithm for global optimization problems. *Journal of Global Optimization*, 62(2): 205–227, Jun 2015. ISSN 1573-2916. doi: 10.1007/s10898-014-0241-8. URL https://doi.org/10.1007/s10898-014-0241-8.

[Liu et al., 2017] Qunfeng Liu, Guang Yang, Zhongzhi Zhang, & Jinping Zeng. Improving the convergence rate of the direct global optimization algorithm. *Journal of Global Optimization*, 67(4): 851–872, Apr 2017. ISSN 1573-2916. doi: 10.1007/s10898-016-0447-z. URL https://doi.org/10.1007/s10898-016-0447-z.

[Liu & Wang, 2015] Yan Y. Liu & Shaowen Wang. A scalable parallel genetic algorithm for the generalized assignment problem. *Parallel Computing*, 46: 98 – 119, 2015. ISSN 0167-8191. doi: https://doi.org/10.1016/j.parco.2014.04.008. URL http://www.sciencedirect.com/science/article/pii/S0167819114000519.

[Loukil et al., 2007] Taïcir Loukil, Jacques Teghem, & Philippe Fortemps. A multi-objective production scheduling case study solved by simulated annealing. *European Journal of Operational Research*, 179(3): 709 – 722, 2007. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.03.073. URL http://www.sciencedirect.com/science/article/pii/S0377221705007319.

[Lozano et al., 2011] Manuel Lozano, Daniel Molina, & Francisco Herrera. Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 15(11): 2085–2087, nov 2011.

[Lu et al., 2019] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, & Wolfgang Banzhaf. Nsga-net: Neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pp. 419–427, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-6111-8. doi: 10.1145/3321707.3321729. URL http://doi.acm.org/10.1145/3321707.3321729.

[Luo et al., 2019] Jia Luo, Shigeru Fujimura, Didier El Baz, & Bastien Plazolles. Gpu based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem. *Journal of Parallel and Distributed Computing*, 133: 244 – 257, 2019. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2018.07.022. URL http://www.sciencedirect.com/science/article/pii/S0743731518305628.

[López Jaimes & Zapotecas-Martínez, 2011] Antonio López Jaimes & Saúl Zapotecas-Martínez. An introduction to multiobjective optimization techniques. *Optimization in Polymer Processing*, pp. 29–58, 01 2011.

[M.A. AL-Salami, 2009] Nada M.A. AL-Salami. Evolutionary algorithm definition. *American Journal of Engineering and Applied Sciences*, 2, 04 2009. doi: 10.3844/ajeassp.2009.789.795.

[Metropolis et al., 1953] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, & Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6): 1087–1092, 1953. doi: 10.1063/1.1699114. URL https://doi.org/10.1063/1.1699114.

[Miettinen et al., 2008] Kaisa Miettinen, Francisco Ruiz, & Andrzej Wierzbicki. *Introduction to Multiobjective Optimization: Interactive Approaches*, pp. 27–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[Molina et al., 2011] Daniel Molina, Manuel Lozano, Ana M Sánchez, & Francisco Herrera. Memetic algorithms based on local search chains for large scale continuous optimisation problems: MA-SSW-Chains. *Soft Computing*, 15(11): 2201–2220, 2011.

[Moore & Bierbaum, 1979] Ramon E. Moore & Fritz Bierbaum. *Methods and Applications of Interval Analysis (SIAM Studies in Applied and Numerical Mathematics) (Siam Studies in Applied Mathematics, 2.)*. Soc for Industrial & Applied Math, 1979. ISBN 0898711614.

[Nakib et al., 2017] Amir Nakib, Salma Ouchraa, Nadyia. Shvai, Léo Souquet, & El-Ghazali Talbi. Deterministic metaheuristic based on fractal decomposition for large-scale optimization. *Applied Soft Computing*, 61(Supplement C): 468 – 485, 2017. ISSN 1568-4946.

[Nakib et al., 2018] Amir Nakib, L. Souquet, & El-Ghazali Talbi. Parallel fractal decomposition based algorithm for big continuous optimization problems. *Journal of Parallel and Distributed Computing*, 2018. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2018.06.002.

[Nebro et al., 2015] Antonio Nebro, Juan Durillo, & Matthieu Vergne. Redesigning the jmetal multi-objective optimization framework. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pp. 1093–1100, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3488-4. doi: 10.1145/2739482.2768462. URL http://doi.acm.org/10.1145/2739482.2768462.

[Nesterov, 1983] Yu. E. Nesterov. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. 1983.

[Neumaier, 1991] Arnold Neumaier. *Interval Methods for Systems of Equations*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1991. doi: 10.1017/CBO9780511526473.

[Nocedal & Wright, 2006] Jorge. Nocedal & Stephen J. Wright. *Numerical optimization*. Springer, 2006.

[Nowak et al., 2014] Krzysztof Nowak, Marcus Märtens, & Dario Izzo. Empirical performance of the approximation of the least hypervolume contributor. vol. 8672, 09 2014.

[Okabe et al., 2003] Tatsuya Okabe, Yaochu Jin, & Bernhard Sendhoff. A critical survey of performance indices for multi-objective optimisation. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, vol. 2, pp. 878–885 Vol.2, Dec 2003. doi: 10.1109/CEC.2003.1299759.

[Paulavičius et al., 2014] Remigijus Paulavičius, Yaroslav D. Sergeyev, Dmitri E. Kvasov, & Julius Žilinskas. Globally-biased disimpl algorithm for expensive global optimization. *Journal of Global Optimization*, 59(2): 545–567, Jul 2014. ISSN 1573-2916. doi: 10.1007/s10898-014-0180-4. URL https://doi.org/10.1007/s10898-014-0180-4.

[Pires et al., 2012] Dulce Fernão Pires, Carlos Henggeler Antunes, & António Gomes Martins. Nsga-ii with local search for

a multi-objective reactive power compensation problem. *International Journal of Electrical Power & Energy Systems*, 43(1): 313 – 324, 2012. ISSN 0142-0615. doi: https://doi.org/10.1016/j.ijepes.2012.05.024. URL http://www.sciencedirect.com/science/article/pii/S0142061512002141.

[Pontes et al., 2016] Fabricio Pontes, Gabriela Amorim, Pedro Balestrassi, Anderson Paiva, & João Ferreira. Design of experiments and focused grid search for neural network parameter optimization. *Neurocomputing*, 186: 22 – 34, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2015.12.061. URL http://www.sciencedirect.com/science/article/pii/S0925231215020184.

[Price et al., 2005] Kenneth. Price, Rainer Storn, & Jouni Lampinen. Differential evolution - a practical approach to global optimization. In *Springer*, 2005.

[Qin & Suganthan, 2005] A K Qin & P N Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785–1791 Vol. 2, sep 2005.

[Ratz & Csendes, 1995] Dietmar Ratz & Tibor Csendes. On the selection of subdivision directions in interval branch-and-bound methods for global optimization. *Journal of Global Optimization*, 7(2): 183–207, Sep 1995. ISSN 1573-2916. doi: 10.1007/BF01097060. URL https://doi.org/10.1007/BF01097060.

[Real et al., 2017] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, & Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2902–2911. JMLR.org, 2017. URL http://dl.acm.org/citation.cfm?id=3305890.3305981.

[Real et al., 2018a] Esteban Real, Alok Aggarwal, Yanping Huang, & Quoc V. Le. Regularized evolution for image classifier architecture search. 2018. URL https://arxiv.org/pdf/1802.01548.pdf.

[Real et al., 2018b] Esteban Real, Alok Aggarwal, Yanping Huang, & Quoc V. Le. Regularized evolution for image classifier architecture search. 2018. URL https://arxiv.org/pdf/1802.01548.pdf.

[Rechenberg, 1965] Ingo Rechenberg. Cybernetic solution path of an experimental problem. 1965.

[Riquelme-Granada et al., 2015] Nery Riquelme-Granada, Christian Von Lücken, & Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pp. 1–11, Oct 2015. doi: 10.1109/CLEI.2015.7360024.

[Ryoo & Sahinidis, 1995] Hong Seo Ryoo & Nick Sahinidis. Global optimization of nonconvex nlps and minlps with applications in process design. *Computers and Chemical Engineering*, 19(5): 551 – 566, 1995. ISSN 0098-1354. doi: https://doi.org/10.1016/0098-1354(94)00097-2. URL http://www.sciencedirect.com/science/article/pii/0098135494000972.

[Saborido Infantes et al., 2017] Rubén Saborido Infantes, Ana Belen Ruiz, & Mariano Luque. Global WASF-GA: An evolutionary algorithm in multiobjective optimization to approximate the whole pareto optimal front. *Evolutionary Computation*, 25(2): 309–349, 2017.

[Sciuto et al., 2019] Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, & Mathieu Salzmann. Evaluating the search phase of neural architecture search. 02 2019.

[Ser et al., 2019] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai N. Suganthan, Carlos A. Coello Coello, & Francisco Herrera. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*, 48: 220 – 250, 2019. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2019.04.008. URL http://www.sciencedirect.com/science/article/pii/S2210650218310277.

[Siarry et al., 1997] Patrick Siarry, Gerard Berthiau, François Durbin, & Jacques Haussy. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. *ACM Trans. Math. Softw.*, 23: 209–228, 06 1997. doi: 10.1145/264029.264043.

[Sierra & Coello Coello, 2005] Margarita Reyes Sierra & Carlos A. Coello Coello. Improving pso-based multi-objective optimization using crowding, mutation and ϵ-dominance. In Carlos A. Coello Coello, Arturo Hernández Aguirre, & Eckart Zitzler, editors, *Evolutionary*

Multi-Criterion Optimization, pp. 505–519, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31880-4.

[Simonyan & Zisserman, 2014] Karen Simonyan & Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[Sinha et al., 2014] Ankur Sinha, Pekka Malo, Peng Xu, & Kalyan Deb. A bilevel optimization approach to automated parameter tuning. *GECCO 2014 - Proceedings of the 2014 Genetic and Evolutionary Computation Conference*, 07 2014. doi: 10.1145/2576768.2598221.

[Situ et al., 2017] Xin Situ, W. N. Chen, Y. J. Gong, Ying Lin, Wei-Jie Yu, Zhiwen Yu, & J. Zhang. A parallel ant colony system based on region decomposition for taxi-passenger matching. In *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 960–967, June 2017.

[Solis & Wets, 1981] Francisco J. Solis & Roger J.-B. Wets. Minimization by Random Search Techniques. *Mathematics of Operations Research*, 6 (1): 19–30, February 1981.

[Srinivas & Deb, 1994] N. Srinivas & Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3): 221–248, September 1994. ISSN 1063-6560. doi: 10.1162/evco.1994.2.3.221. URL http://dx.doi.org/10.1162/evco.1994.2.3.221.

[Srivastava et al., 2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, & Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15: 1929–1958, 06 2014.

[Steuer & Choo, 1983] Ralph E. Steuer & Eng-Ung Choo. An interactive weighted tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(3): 326–344, Oct 1983. ISSN 1436-4646. doi: 10.1007/BF02591870. URL https://doi.org/10.1007/BF02591870.

[Stewart et al., 2008] Theodor Stewart, Oliver Bandte, Heinrich Braun, Nirupam Chakraborti, Matthias Ehrgott, Mathias Göbelt, Yaochu Jin, Hirotaka Nakayama, Silvia Poles, & Danilo Di Stefano. *Real-World Applications of Multiobjective Optimization*, pp. 285–327.

Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-88908-3. doi: 10.1007/978-3-540-88908-3_11. URL https://doi.org/10.1007/978-3-540-88908-3_11.

[Storn & Price, 1995] Rainer Storn & Kenneth Price. *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*, vol. 3. ICSI Berkeley, 1995.

[Suganuma et al., 2017] Masanori Suganuma, Shinichi Shirakawa, & Tomoharu Nagao. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pp. 497–504, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4920-8. doi: 10.1145/3071178.3071229. URL http://doi.acm.org/10.1145/3071178.3071229.

[Szegedy et al., 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, & Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594.

[Talbi, 2009] El-Ghazali Talbi. *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[Tang et al., 2007] Ke Tang, X\in Yáo, Ponnuthurai Nagaratnam Suganthan, Cara MacNish, Ying-Ping Chen, Chih-Ming Chen, & Zhenyu Yang. Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. *Nature Inspired Computation and Applications Laboratory, USTC, China*, pp. 153–177, 2007.

[Tawarmalani & Sahinidis, 2004] Mohit Tawarmalani & Nikolaos Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Math. Program.*, 99: 563–591, 04 2004. doi: 10.1007/s10107-003-0467-6.

[Tseng & Chen, 2008] Lin-Yu Tseng & Chun Chen. Multiple trajectory search for Large Scale Global Optimization. In *2008 IEEE Congress on Evolutionary Computation*, pp. 3052–3059, June 2008.

[Tuy & Horst, 1996] Hoang Tuy & Reiner Horst. *Global Optimization: Deterministic Approaches*. Springer, 1996.

[Varelas et al., 2018] Konstantinos Varelas, Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Ouassim Ait Elhara, Yann Semet, Rami Kassab, & Frederic Barbaresco. *A Comparative Study of Large-Scale Variants of CMA-ES: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part I*, pp. 3–15. 01 2018. ISBN 978-3-319-99252-5. doi: 10.1007/978-3-319-99253-2_1.

[Vazan & Cervenanska, 2018] Pavel Vazan & Zuzana Cervenanska. Comparison of the scalarization approaches in many-objective simulation-based optimization in production system control. In *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, vol. 1, pp. 56–59, Sep. 2018. doi: 10.1109/STC-CSIT.2018.8526670.

[Vesterstrom & Thomsen, 2004] Jakob Vesterstrom & René Thomsen. A comparative study of differential evolution, particle swarm optimization and evolutionary algorithms on numerical benchmark problems. In *In In Proc. of IEEE Congress on Evolutionary Computation 2004 (CEC'2004)*, pp. 1980–1987, Portland, Oregon, USA, June 20-23 2004.

[Wang & Li, 2009] Yu Wang & Bin Li. A self-adaptive mixed distribution based uni-variate estimation of distribution algorithm for large scale global optimization. In Raymond Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, pp. 171–198. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[Wang et al., 2013] Yu Wang, Jin Huang, Wei Shan Dong, Jun Chi Yan, Chun Hua Tian, Min Li, & Wen Ting Mo. Two-stage based ensemble optimization framework for large-scale global optimization. *European Journal of Operational Research*, 228(2): 308–320, 2013.

[Whitley et al., 1995] Darrell Whitley, Ross Beveridge, Christopher Graves, & Keith Mathias. Test driving three 1995 genetic algorithms: New test functions and geometric matching. *Journal of Heuristics*, 1(1): 77–104, 1995. ISSN 1572-9397.

[Wistuba et al., 2019] Martin Wistuba, Ambrish Rawat, & Tejaswini Pedapati. A survey on neural architecture search. *ArXiv*, abs/1905.01392, 2019.

[Xie & Yuille, 2017] Lingxi Xie & Alan Yuille. Genetic cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1388–1397, Oct 2017. doi: 10.1109/ICCV.2017.154.

[Yang et al., 2007] Guo-Ping Yang, San-Yang Liu, Jianke Zhang, & Quan-Xi Feng. Control and synchronization of chaotic systems by differential evolution algorithm. *Chaos, Solitons and Fractals*, 34(2): 412–419, 2007.

[Zhang & Li, 2007] Qingfu Zhang & Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6): 712–731, Dec 2007.

[Zitzler, 1999] Eckart Zitzler. Evolutionary algorithms for multiobjective optimization: Methods and applications, 1999.

[Zitzler & Thiele, 1998] Eckart Zitzler & Lothar Thiele. Multiobjective optimization using evolutionary algorithms — a comparative case study. In Agoston E. Eiben, Thomas Bäck, Marc Schoenauer, & Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, pp. 292–301, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[Zitzler & Thiele, 1999] Eckart Zitzler & Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3 (4): 257–271, Nov 1999. ISSN 1089-778X. doi: 10.1109/4235.797969.

[Zitzler et al., 2000] Eckart Zitzler, Kalyanmoy Deb, & Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2): 173–195, 2000. doi: 10.1162/106365600568202.

[Zoph & Le, 2016] Barret Zoph & Quoc Le. Neural architecture search with reinforcement learning. 11 2016.

[Zoph et al., 2018] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, & Quoc V. Le. Learning transferable architectures for scalable image

recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8697–8710, 06 2018. doi: 10.1109/CVPR.2018.00907.