UNIVERSITY OF VERACRUZ

DOCTORAL THESIS

Memetic Differential Evolution for Constrained Numerical Optimization Problems

Author:
Saúl DOMÍNGUEZ-ISIDRO

Supervisor:
Dr. Efrén MEZURA-MÓNTES

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy on Artificial Intelligence

in the

Artificial Intelligence Research Center

December 12, 2017

University of Veracruz

Abstract

Artificial Intelligence Research Center

Doctor of Philosophy on Artificial Intelligence

Memetic Differential Evolution for Constrained Numerical Optimization Problems

by Saúl DOMÍNGUEZ-ISIDRO

The object of study of this dissertation is Memetic Differential Evolution Algorithms (MDEs) for constrained numerical optimization problems (CNOPs). MDEs are one of the most used approaches to improve the performance of the standard differential evolution (DE) for the solution of numerical optimization problems, which are present in real-world applications, often in engineering problems. As it is well known, memetic approaches are characterized by the inclusion of search operators within the cycle of an evolutionary algorithm, improving the search process on a broader range of problems, due to the synergy between the global and local search operator. However, the coordination among the algorithmic components is an issue in the design part of a memetic algorithm, since the excessive use of the local search operator could affect the efficiency of the algorithm. Therefore, three main studies of the effects of local search operators in MDE schemes are carried out. The first study analyzes the relationship between the performance of the local search operator within an MDE and its final results in CNOPs by adopting an improvement index measure, which indicates the rate of fitness improvement made by the local search operator. The second study analyzes the influence of the depth of direct local search methods within MDE when solving CNOPs. Finally, the third study analyzes the Baldwin effect and Lamarckian learning on an MDE that solves CNOPs. Derived from the assessments mentioned above, we propose a coordination mechanism of multiple local search operators for a multimeme scheme based on Differential Evolution (MmDE) that solves CNOPs. The proposed approach associates a pool of direct local search operators within the standard Differential Evolution. The coordination mechanism consists of a probabilistic method based on a cost-benefit scheme, and it is aimed to regulate the activation probability of every local search operator during the evolutionary cycle of the global search. For all implementations, the ε-constrained method is used as constraint-handling technique. All experiments are tested on thirty-six well-known benchmark problems.

Acknowledgements

I would like to acknowledge my thesis director Dr. Efrén Mezura Montes for all the support provided during the doctoral program.

I also want to thank the reviewers for this document for their comments and suggestions which helped to enhance its quality. Thank you to Dr. Guillermo de Jesús Hoyos Rivera, Dr. Guillermo Leguizamón, Dra. Marcela Quiroz Castellanos, Dr. Edgar Alfredo Portilla Flores and Dr. Alejandro Raúl Hernández Montoya.

I want to thank Dr. Ferrante Neri for receiving me during the doctoral stage at De Montfort University of Leicester, UK.

Finally, I want to acknowledge support from CONACYT through project No. 220522, and the University of Veracruz to pursue PhD studies.

Contents

List of Figures

List of Tables

To my beloved wife and my little son Liam

Chapter 1

Introduction

This chapter presents the aim and motivation of this dissertation, which consists of the study of memetic algorithms based on Differential Evolution for solving constrained numerical optimization problems. Likewise, the contributions of this research, the publications generated and the document structure are described.

1.1 Motivation

Constrained numerical optimization problems (CNOPs), also known as the nonlinear general problem [1], consists of finding a solution, among a set of solutions, that optimizes an objective function and satisfies a series of constraint functions [2].

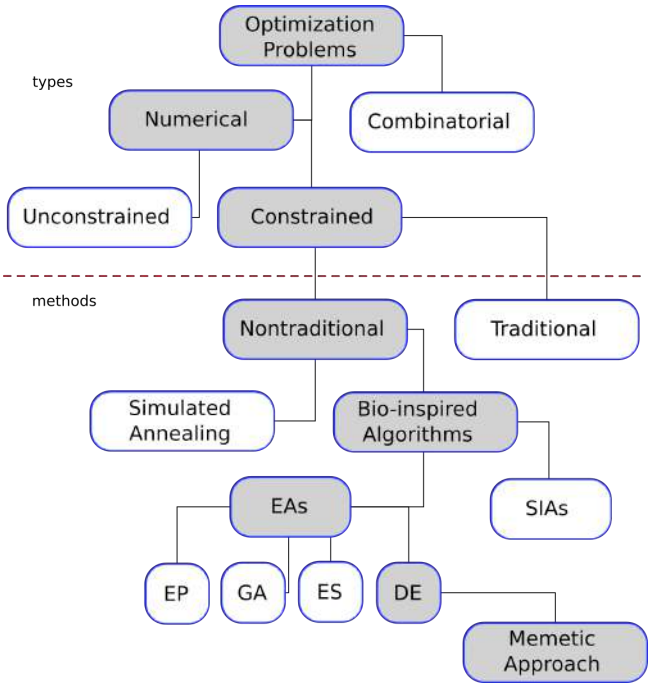
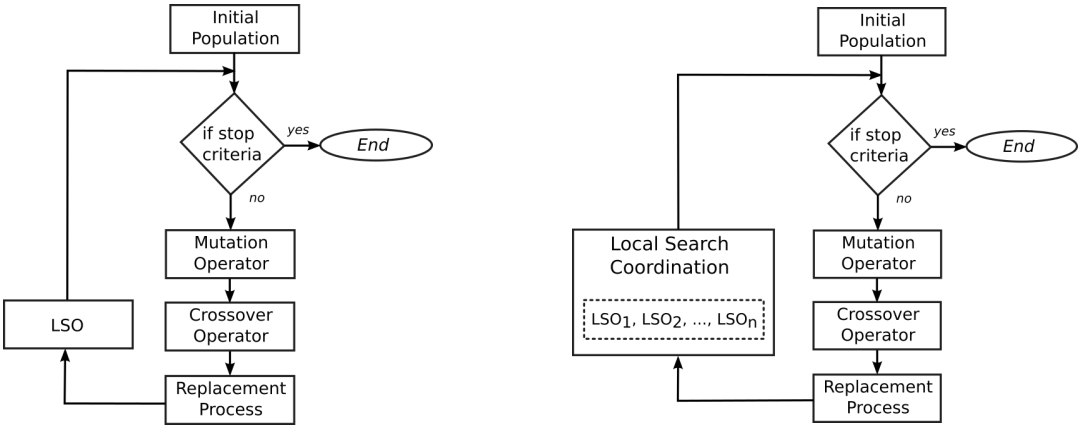


FIGURE 1.1: Classification of optimization problems and methods to tackle them. Gray boxes highlight the problem and methods treated in this work. The abbreviations, EAs, SIAs, EP, GA, ES, and DE, mean evolutionary algorithms, swarm intelligence algorithms, evolutionary programming, genetic algorithm, evolution strategy and, differential evolution, respectively.

CNOPs are frequently present in different disciplines such as engineering, mechatronic, bio-informatics, management, etc. [3, 4].

There are several methods that solve CNOPs, and can be divided as traditional and nontraditional methods [5]. Whereas, the traditional methods allow to solve CNOPs under particular conditions of the functions and constraints, nontraditional methods can be adapted to solve a wider range of CNOPs. Although both classes of methods are capable of solving CNOPs, nontraditional methods are usually occupied in problems with a greater number of constraints and functions with a high dimensionality. Therefore, this thesis is focused on the study of a type of algorithm belonging to nontraditional methods, see Figure 1.1.

Differential Evolution (DE), introduced by Storn and Price in [6], is one of the most popular and efficient evolutionary algorithms (EAs). DE has been applied to solve different optimization problems in diverse fields [7] including CNOPs. However, coupled with No Free Lunch Theorems (NFLT) [8], the complexity and diversity of problems have allowed the emergence of more sophisticated approaches within the DE variants including memetic approaches. A memetic approach consists in the interaction of two main techniques, global and local search [9] to improve the efficiency and reduce the limitations in the search logic, such as the lack of search movements [10, 11] by obtaining the advantage of both procedures. Despite the fact that there are studies that prove the benefits of local search operators (LSO) within evolutionary algorithms [12], finding a suitable global-local search interaction is an inherent challenge in the design phase of memetic algorithms (MAs), and it depends mainly on both, the problem to solve and the global and local search features [13].



(A) Generic MDE (B) Generic MmDE

FIGURE 1.2: Generics Memetic and Multimeme Differential Evolution (a) and (b), respectively. LSO means local search operator. Dotted box indicates an optional process.

Different approaches of Memetic Differential Evolution (MDE) for CNOPs can be found in the literature, see Fig. 1.2(a), which includes those schemes where the LSO is based on gradient methods [14, 15] based on direct methods [16, 17, 18, 19] and special operators [20, 21, 22]. Likewise, in [23] a MDE approach is based on a Covariance Matrix Adaptation Evolution Strategy to coordinate the global and local search methods. The approaches above present different types of coordination between the LSO and different Differential Evolution variants, and also report competitive results. Nevertheless, for *Multimeme* Differential Evolution (MmDE), term adopted

by Krasnogor and Smith in [24] referring memetic algorithms with multiple LSOs, the coordination among the components becomes more complex (see Fig. 1.2(b)), since a trade-off among LSOs must be considered in order to avoid negative effects of LSOs during the global search process [25]. There are different works which have studied the coordination of multiple LSOs within Multimeme Differential Evolution (MmDE), such as in [10, 26, 27, 28, 29]. In [30] the authors proposed a MmDE algorithm which, in addition to coordinating multiple LSOs, it is also capable of coordinating different crossover and mutation operators with a pool of parameter values. However, all works above were designed for unconstrained optimization problems.

On the other hand, there are just a few studies in the coordination of local search methods for CNOPs, such as in, [31, 32], which a pool of LSOs is coordinated based on their performance over an agent-based memetic approach. However, to the best of the author's knowledge coordination methods to handle multiple LSOs within a multimeme scheme based on DE has not been considered for CNOPs. One of the possible reasons for the lack of DE-based multimeme approaches for CNOPs is the complexity of handling constraints, and the DE versatility, such as in [33] where an ensemble of constraint handling techniques was proposed to maximize the efficiency of the search process. Therefore, finding a suitable balance among the algorithm components (DE, constraint handler and LSOs) implies a challenge.

Therefore, this dissertation is aimed to design a Multimeme Differential Evolution scheme derived from an empirical study, which exposes the local search operators benefits within Differential Evolution for CNOPs.

1.2 Problem Statement

A constrained numerical optimization problem (CNOP), also called constrained nonlinear optimization problem [34], can be found in real-world problems and is defined, without loss of generality, as follows:

Minimize

$$f(X), \qquad X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^n \tag{1.1}$$

Subject to

$$g_i(X) \leq 0, \qquad i = 1, \ldots, m$$

$$h_j(X) = 0, \qquad j = 1, \ldots, p$$

$$L_k \leq x_k \leq U_k \qquad x_k \in \mathbb{R}$$

where X is the vector of solutions, m is the number of inequality constraints, and p is the number of equality constraints. L_k and U_k define the lower and upper limits on dimension k, respectively, of the search space $\mathcal{S}$.

In CNOPs, the feasible region $\mathcal{F}$ is the set of all solutions which satisfy the constraint functions ($g_i(X)$ and $h_i(X)$). Usually, equality constraints are transformed into inequality constraints as follows [35]: $|h_j(X)| - \delta \leq 0$, where δ is the tolerance allowed.

1.3 Hypothesis

Based on an empirical study about the benefits of differents LSOs, we can design a MmDE to solve CNOPs, which will have a highly competitive performance with respect to state-of-the-art algorithms.

1.4 Goal

Significantly advance the knowledge of Memetic Algorithms in constrained numerical optimization problems and operations-research area, by designing a Multimeme Differential Evolution scheme derived from an empirical study, which exposes the local search operators benefits within Differential Evolution.

1.5 Contributions

1. A detailed empirical study of the benefits of different types of local search operators within Differential Evolution for CNOPs.
2. An efficient local search coordination mechanism for a multimeme Differential Evolution scheme for CNOPs.

Derived from this research four publications were obtained, one of which was published in a specialized journal in the area of evolutionary computation, and three were published in the proceedings of international conferences.

- S. Domínguez-Isidro, E. Mezura-Montes, **A cost-benefit local search coordination in multimeme differential evolution for constrained numerical optimization problems**. Swarm and Evolutionary Computation, October 2017, ISSN 2210-6502, https://doi.org/10.1016/j.swevo.2017.10.006. IF:3.893
- S. Dominguez-Isidro and E. Mezura-Montes. **The baldwin effect on a memetic differential evolution for constrained numerical optimization problems**. In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '17). ACM, New York, NY, USA, 203-204. DOI: https://doi.org/10.1145/3067695.3076096
- S. Domínguez-Isidro, E. Mezura-Montes, **Study of Direct Local Search Operators Impact in Memetic Differential Evolution for Constrained Numerical Optimization Problems**. In the 27th International Conference on Electronics, Communications, and Computers, CONIELECOMP 2017.
- S. Dominguez-Isidro, E. Mezura-Montes, and G. Leguizamon, **Performance Comparison of Local Search Operators in Differential Evolution for Constrained Numerical Optimization Problems**. 2014 IEEE Symposium on Differential Evolution (SDE), Orlando, FL, USA, pp. 1–8.

1.6 Document Structure

This dissertation has been divided in eigth Chapters, which are described bellow:

- **Chapter 1. Introduction.** Describes the motivation, scope and goals of this dissertation.
- **Chapter 2. Fundamentals of Mathematical Programming.** The theoretical foundations of optimization problems are described in this chapter. As well as the main traditional and non-traditional methods to address this type of problems.
- **Chapter 3. Bio-inspired Algorithms.** This chapter describes the most important paradigms of search algorithms inspired by nature. Also, the methods for handling constraints are described.
- **Chapter 4. Memetic Differential Evolution for Constrained Problems: Syntactic Model and Taxonomy.** This chapter describes the fundamentals of memetic algorithms as well as state-of-the-art memetic algorithms, which are based on DE for solving constrained numerical optimization problems.
- **Chapter 5. Local Search Operators Influence in Memetic DE Approach for CNOPs.** Three empirical studies measuring relevant aspects of local search operators within memetic algorithms based on DE to solve CNOPs are presented in this chapter.
- **Chapter 6. A Multimeme Differential Evolution Framework.** In this chapter, we describe our proposal, which consists of a mechanism of memes coordination based on the cost-benefit of local search operator, for multimeme algorithms based on DE.
- **Chapter 7. Conclusions and Future Work.** Finally, this chapter presents the conclusions of the thesis and defines the final remarks of this research.

Chapter 2

Fundamentals of Mathematical Programming

This chapter presents the main aspects of addressing a nonlinear optimization problem. Also, some direct methods (traditional and non-traditional) to solve this type of problems are described.

2.1 Optimal Problem Formulation

The optimal problem formulation is the procedure where a real-world problem is modeled as a set of mathematical functions to be optimized. The assumed real-world is abstracted from the real case, concentrating it on the main variables that control the behavior of the real system. The model, as it is an abstraction of the supposed real-world, expresses in an adequate form the mathematical functions that represent the behavior of the supposed system [36], see Figure 2.1.

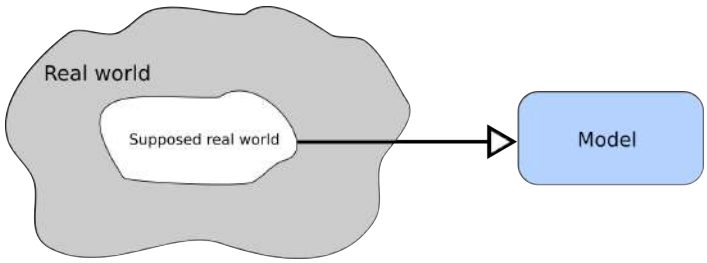


FIGURE 2.1: Levels of abstraction in the model development

Once the mathematical model of the problem to be optimized has been formulated, it must be solved using an optimization algorithm. However, there are four important aspects to consider in formulating the optimization problem:

1. **Decision variables.** Also known as design variables, they are a set of values that define the mathematical model of the problem. They are also used to determine the objective function and constraints of the model.

2. **Constraints.** Represent some functional relationships among the decision variables and other design variables satisfying certain physical phenomenom and certain resource limitations. Constraints can be divided in two groups: Inequality and Equality constraints. Whereas the first ones state that the functional relationships among decision variables are either greater than, smaller

than, or equal to, a resource value, the Equality constraints state that the functional relationships should exactly match a resource value.

3. **Objective function.** It is a mathematical relationship between decision variables, parameters and a magnitude that represents the objective or product of the system. The objective function can be maximized or minimized.
4. **Variable bounds.** They define the search space of the problem since they set the upper and lower frontiers for each decision variable.

Having considered the four aspects above, the optimization problem can be mathematically defined in a specific format, known as nonlinear programming, which is defined in Section 1.2 of this work.

2.2 Optimality Criteria

In an optimization problem, three different types of optimal points can be defined. We consider as a point a set of decision variables for a given objective function.

Local optimal point: A solution X^* is said to be a local optimal point, if there exists no point in the neighborhood of X^* which is better than X^*. For minimization problems, a point X^* is a locally minimal point if no point in the neighborhood has a function value smaller than $f(X^*)$ [5].

Global optimal point: A solution X^{**} is said to be a global optimal point, if there exists no point in the entire search space which is better than de point X^{**}. Similarly, a point X^{**} is a global minimal point if no point in the entire search space has a function value smaller than $f(X^{**})$ [5].

Inflexion point: A solution X^* is said to be an inflection point if the function value increases locally as X^* increases and decreases locally as X^* decreases, or if its function value decreases locally as X^* increases and increases locally as X^* decreases [5].

Once the different types of optimal points have been defined, the optimality criteria differ, depending on the number of decision variables of the objective function (single-variable and multivariable optimization problems).

2.2.1 Single-Variable Optimization Problems

As its name indicates, the optimization problem only works with a single decision variable, so that particular characteristics of the objective function can be exploited to verify if a point is a local minimum or a global maximum or a point of inflection.

Assuming that the first and the second-order derivatives of the objective function $f(x)$ exist in the chosen search space, the function can be expanded in Taylor's series at any point $\bar{x}$ and satisfy the condition that any other point in the vecinity has a larger function value. It can then be shown that conditions for a point $\bar{x}$ to be a minimum point is that $f'(\bar{x}) = 0$ and $f''(\bar{x}) > 0$, where f' and f'' represent the first and second derivatives of the function [5].

The first condition alone suggests that the point is either a minimum, a maximum or an inflection point, and both conditions together indicate that the point is a minimum. In general, the sufficient conditions of optimality are given as follows:

Suppose at point x^*, the first derivative is zero and the first nonzero higher order derivative is denoted by n; then

- If n is odd, x^* is an inflection point.
- If n is even, x^* is a local optimum.
    1. If the derivative is positive, x^* is a local minimum.
    2. If the derivative is negative, x^* is a local maximum.

2.2.2 Multivariable Optimization Problems

As in single-variable functions, the definitions of local, global and an inflection point are the same for multivariate functions. However, the criterion of optimality substantially changes. In a multivariable function, the gradient of a function is not a scalar quantity; instead it is a vector [5].

Considering that the objective function is a function of N variables represented by $X = [x_1, x_2, \ldots, x_N]$. The gradient vector at any point $X^{(t)}$ is represented by Equation 2.1:

$$\nabla f(X^{(t)}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_N} \right)^T \tag{2.1}$$

On the other hand, the second-order derivatives in multivariable functions form a matrix, better known as the Hessian matrix, given as follows:

$$\nabla^2 f(X^{(t)}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \frac{\partial^2 f}{\partial x_N \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix} \tag{2.2}$$

Then, the optimality criteria are described bellow:

A point $\bar{X}$ is a stationary point if $\nabla f(\bar{X}) = \mathbf{\emptyset}$. Furthermore, the point is a minimum, a maximum, or an inflexion point if $\nabla^2 f(\bar{X})$ is positive-definite, negative-definite, or otherwise [5].

A matrix $\nabla^2 f(X^{(t)})$ is defined to be positive-definite if for any point y in the search space the quantity $y^T \nabla^2 f(X^{(t)}) y \geq 0$. The matrix is called a negative definite matrix if at any point the quantity $y^T \nabla^2 f(X^{(t)}) y \leq 0$. if at some point y^+ in the search space the quantity $y^{+T} \nabla^2 f(X^{(t)}) y^+$ is positive and at some other point y^- the quantity $y^{-T} \nabla^2 f(X^{(t)}) y^-$ is negative, then the matrix $\nabla^2 f(X^{(t)})$ is neither positive-definite

nor negative-definite. Other way to test the positive-definiteness of a matrix is to calculate the principal determinants of the matrix. If all principal determinants are positive, the matrix is positive-definite. It is worth mentioning here that the negative-definiteness of a matrix A can be verified by testing the positive-definiteness of the matrix $-A$ [5].

2.3 Kuhn-Tucker Conditions

The Kuhn-Tucker (K-T) conditions determine whether a point (solution) is a candidate to be the optimun of a constrained problem, see Section 1.2. Considering the *Nonlinear Programming Problem* (NLP) defined in Equation 1.1, the inequality and equality constraints can be added to the objective function to form an unconstrained problem by using Lagrange multiplier technique. In this way, K-T conditions can be obtained by satisfying the first-order optimality conditions of that unconstrained problem [5]:

$$\nabla f(X) - \sum_{j=1}^{J} u_j \nabla g_j(X) - \sum_{k=1}^{K} v_k \nabla h_k(X) = 0, \tag{2.3}$$

$$g_j(X) \geq 0, j = 1, 2, \ldots, J; \tag{2.4}$$

$$h_k(X) = 0, k = 1, 2, \ldots, K; \tag{2.5}$$

$$u_j g_j(X) = 0, j = 1, 2, \ldots, J; \tag{2.6}$$

$$u_j(X) \geq 0, j = 1, 2, \ldots, J; \tag{2.7}$$

Because the multiplier u_j corresponds to the j-th inequality constraint and the multiplier v_k corresponds to the k-th equality constraint, there are a total number of J entries in the u-vector and K entries in the v-vector. Equation 2.3 arises from the optimality condition of the unconstrained Lagrangian function. Equations 2.4 and 2.5 are required for satisfying the constraints. Equation 2.6 arises only for inequality constraints. If the j-th inequality constrait is active at a point X (that is, $g_j = 0$), the product $u_j g_j(X) = 0$. On the other hand, if an inequality constraint is inactive at a point X (that is, $g_j > 0$), the Lagrange multiplier u_j is equal to zero, meaning that in the neighborhood of the point X the constraint has no effect on the optimal point. The final inequality condition suggests that in the case of active constraints the corresponding Lagrange multiplier must be positive.

Because K-T conditions are applied just for inequality constraints, the variable bounds $x_i^{(L)} \leq x_i \leq x_i^{(U)}$ should be considered as inequality constraints, an can be written in two inequality constraints as follows:

$$x_i - x_i^{(L)} \geq 0, \tag{2.8}$$

$$x_i^{(U)} - x_i \geq 0. \tag{2.9}$$

To verify whether a point is a K-T point, all the aforementioned conditions are expressed in terms of u and v vectors. If there exists at least one set of u and v vectors, which satisfy all K-T conditions the point is said to be a K-T point.

Kuhn-Tucker necessity theorem[5]

Consider the NLP problem in Equation 1.1. Let f , g_j and h_k be differentiable functions and X^* be a feasible solution to the NLP. Let $I = j|g_j(X^*) = 0$ denote the set of active inequality constraints. Furthermore, $\nabla g_j(X^*)$ for $j \in I$ and $\nabla h_k(X^*)$ for $k = 1, 2, \ldots, K$ are linearly independent (known as constraint qualification). If X^* is an optimal solution to the NLP problem, there exists a (u^*, v^*), such that (X^*, u^*, v^*) satisfies the K-T conditions.

If a feasible point satisfies the constraint qualification condition, the K-T necessity theorem can be used to prove that the point is not optimal. However, if the constraint qualification is not satisfied at any point, the point can or can not be an optimal point. Thus, this theorem can only be used to conclude whether a point is not an optimal point.

One of the main drawbacks to verify the constraint qualification is the fact that the global optimum must be known apriori. Likewise, for some NLP problems, the constraint qualification is satisfied when a set of properties in the constraint functions are fulfilled: (1) all the inequality and equality constraints are linear, (2) when all the inequality constraints are concave functions, and the equality constraints are linear and there exists at least one feasible point X that is rigorously inside the feasible region of the inequality constraints.

Kuhn-Tucker sufficiency theorem[5]

Let the objective function be convex, the inequality constraints $g_j(X)$ be all concave functions for $j = 1, 2, \ldots, J$ and equality constraints $h_k(X)$ for $k = 1, 2, \ldots, K$ be linear. If there exists a solution $(X^* u^* v^*)$ that satisfies the K-T conditions, then X^* is an optimal solution to the NLP problem.

The sufficiency conditions require the Hessian matrix of each function to verify if they are positive or negative defined, and often practical problems may not possess these properties, making it difficult to verify.

It is important to note that optimality conditions for constrained problems are valid only when the problem to solve has some very stringent features. Consequently, constrained optimization is an open problem. Therefore, different mathematical programming techniques and also heuristic-based methods have been proposed to solve this type of problems.

2.4 Direct Methods

In this section direct methods to solve multivariable optimization problems are presented. Indirect methods are omitted because this dissertation focuses only on direct methods. Direct methods are also known as order-zero methods since they do not require gradient information of the function to be optimized during the optimization process.

2.4.1 Hooke-Jeeves Method (HJ)

HJ works by creating a set of search directions iteratively. HJ [37] needs three user-defined parameters: (1) the variable increments Δ, (2) a step reduction factor $\alpha > 1$ and (3) the termination criterion $maxFES$. HJ has two types of moves in the search space: the exploratory move and the pattern move. In the exploratory move, the current vector is perturbed in positive and negative directions along each variable one at a time and the best vector is recorded, see Algorithm 1, where $X_{(k)}$ is the vector at iteration k and D is the vector dimension.

Algorithm 1 Exploratory Move

1: $X \leftarrow X_{(k)}$
2: **for** $i \leftarrow 1, D$ **do**
3: Calculate $X \leftarrow (X_i)$, $U \leftarrow (X_i + \Delta)$, $V \leftarrow (X_i - \Delta)$
4: Set $X_i \leftarrow$ getBest(X,U,V)
5: **end for**
6: **if** The new vector is different than the initial vector **then**
7: **return** success
8: **else**
9: **return** failure
10: **end if**

Pattern move is applied to the best vector $X_{(k)}$ at iteration k to find a new vector X_p, see Equation 2.10.

$$X_p = X_{(k)} + (X_{(k)} - X_{(k-1)}) \tag{2.10}$$

The complete local search operator is described in Algorithm 2.

Algorithm 2 Hooke-Jeeves Method

1: Set X_k by exploratory move with Xs using Algorithm 1
2: **repeat**
3: **if** Success **then**
4: **repeat**
5: Perform a pattern move using Equation 2.10
6: Perform an exploratory move with X_p using Algorithm 1.
Let the result be the new vector.
7: **until** New vector is worse than the previous vector
8: **end if**
9: Set $\Delta \leftarrow \Delta/\alpha$
10: Set $X_{(k+1)}$ by exploratory move with $X_{(k)}$ using Algorithm 1
11: **until** *Stop condition*
12: **return** $X_{(k)}$

2.4.2 Nelder-Mead Method (NM)

NM [38] works through expansions and contractions during the main loop (see Equation 2.11), and needs three user-defined parameters: (1) the expansion factor γ, (2) the contraction factor β and (3) the termination criterion $maxFES$.

$$X_{new} = \begin{cases} (1+\gamma)X_c - \gamma X_h \text{ if } f(X_r) < f(X_l) \text{ (expansion)} \\ (1-\beta)X_c + \beta X_h \text{ if } f(X_r) \geq f(X_h) \text{ (contraction)} \\ (1+\beta)X_c - \beta X_h \text{ if } f(X_g) < f(X_r) < f(X_h) \text{(contraction)} \\ X_r \text{ , otherwise} \end{cases} \tag{2.11}$$

where X_h, X_l and X_g are the worst, the best and the next to the worst vector of the simplex population respectively. X_r is the reflected point which is calculated by $X_r = 2X_c - X_h$. Finally X_c is the centroid vector, Equation 2.12.

$$X_c = \frac{1}{D} \sum_{i=1, i \neq h}^{D+1} X_i \tag{2.12}$$

where $D+1$ corresponds to the number of vectors in the simplex array and D is the number of decision variables of each vector, i.e, $D = N$. The whole process is shown in Algorithm 3

Algorithm 3 Nelder-Mead Method

1: Create an initial simplex applying Equation 2.13
2: **repeat**
3: Find X_h, X_l and X_g
4: Calculate X_c using Equation 2.12
5: Calculate the reflected vector $X_r = 2X_c - X_h$
6: Set X_{new} using Equation 2.11
7: Set $X_h \leftarrow X_{new}$
8: **until** *Stop condition*
9: **return** X_l

The initial simplex, in this work, is an array of vectors $S = X_0, X_1, \ldots X_N$ generated from a initial vector Xs, see Equation 2.13:

$$X_{i,j} = \begin{cases} Xs_j + rand[0,1] \text{ if } rand(0,1) \leq 0.5 \\ Xs_j - rand[0,1] \text{ , otherwise} \end{cases} \tag{2.13}$$

2.4.3 Random-Walk Method (RW)

RW [5] works by creating a set of P vectors randomly within a range Z_0, if the new vector outperforms the original vector then the new vector is stored in a list; finally the best of the list is selected and the original vector is replaced. The process is repeated until a maximum number of fitness evaluation $maxFES$ is reached. The range Z_0 is reduced every iteration by a reduction factor ε. RW needs four user-defined parameters: (1) the number of vectors P, (2) an initial range Z_0, (3) a step reduction factor ε and (4) the termination criterion $maxFES$. The complete process is shown in Algorithm 4.

Algorithm 4 Random Walk Method

Set $X_{(k)} \leftarrow Xs$
repeat
  for $i \leftarrow 1$ to P **do**
    Set $X_i \leftarrow X_{(k)}$ + random(-0.5, 0.5) Z_0
    if X_i is better than $X_{(k)}$ **then**
      $List_i \leftarrow X_i$
    end if
  end for
  Set $X_{(k+1)} \leftarrow$ getBest($List$)
  Set $Z_0 \leftarrow (1-\varepsilon)Z_0$
until *Stop condition*

2.4.4 Simulated-Annealing Algorithm (SA)

SA [39] works by generating random vectors iteratively, if a new generated vector is better than the previous vector or by means of a probability then previous vector is replaced. The process is repeated until a maximum number of fitness evaluation $maxFES$ is reached, see Algorithm 5.

$$\Delta = \begin{cases} f(X_{(k+1)}) - f(X_{(k)}) \text{ if } \phi(X_{(k+1)}) \text{ and } \phi(X_{(k)}) > 0 \\ \phi(X_{(k+1)}) - \phi(X_{(k)}) \text{ , otherwise} \end{cases} \tag{2.14}$$

Algorithm 5 Simulated Annealing Algorithm

Set $X_{(k)} \leftarrow Xs$
repeat
  Select $p \leftarrow$ random(1, D)
  Set $\Delta \leftarrow 0$
  Set $X_{(k-1)} \leftarrow X_{(k)}$
  for $i \leftarrow p$ to D **do**
    Set $X_{(k),i} \leftarrow$ random(L_i, U_i)
  end for
  Calculate Δ using Equation 2.14
  if $X_{(k)}$ is better than $X_{(k-1)}$ or random(0, 1)$< \exp(\Delta/T)$ **then**
    $X_{(k+1)} \leftarrow X_{(k)}$
  end if
  $T \leftarrow (1-\varepsilon)T$
  $k \leftarrow k+1$
until *Stop condition*

2.4.5 Hill-Climbing Algorithm (HC)

HC [40] consists of random perturbations of the variables in a vector, if a new vector generated is better than the original vector, it is replaced. The process is repeated until a maximum number of fitness evaluations $maxFES$ (user-defined parameter) is reached. The complete process is shown in Algorithm 6, where Xs is the initial vector, k is the number of iteration, D is the vector dimension, i.e., the number of

decision variables in the vector. Finally p is the vector position (selected randomly) to be modified by a random value $\sigma \in [0, 1]$.

Algorithm 6 Hill Climbing Algorithm

1: Set $X_{(k)} \leftarrow Xs$
2: **repeat**
3: Select $\sigma \leftarrow$ random(0, 1)
4: Select $p \leftarrow$ random(1, D)
5: Set $U \leftarrow X_{(k)}$ and $V \leftarrow X_{(k)}$
6: Calculate $U \leftarrow X_{(k),p} + \sigma$ and $V \leftarrow X_{(k),p} - \sigma$
7: Set $X_{(k+1)} \leftarrow$ getBest($X_{(k)}$,U,V)
8: **until** *Stop condition*
9: **return** $X_{(k)}$

Chapter 3

Nature-inspired Algorithms for Optimization

Nature has been a source of inspiration for solving problems in various areas of study, such as industrial design, architecture, engineering, robotics, either by the interaction of multiple organisms, biological structures or physiological characteristics of numerous species.

In the domain of Artificial Intelligence, Evolutionary Computation (EC) emulates natural processes based on Neo-Darwinism principles [41] to find solutions to complex optimization problems. Within the optimization scope, there is a branch based on mechanisms of cooperation in organisms to obtain a mutual benefit, called Swarm Intelligence (SI) [42] which is also implemented to solve optimization problems.

This chapter presents the main EC and SI paradigms, which are frequently used to solve optimization problems, including constrained numerical optimization problems. Likewise, Differential Evolution is described in more detail than the other meta-heuristics, because it is a fundamental part of this research. Finally, the constraint-handling techniques used in this work are described.

3.1 Evolutionary Algorithms

Evolutionary Algorithms (EAs) [43] are a set of metaheuristics used to solve a vast range of problems, from the optimization of combinatorial and numerical problems, the conception of artifacts, information search, device control, and automatic learning, among others [44]. EAs are based on phenomena related to the evolution of species and the survival of the fittest.

According to the general model of an EA, a population of possible solutions to the problem to be solved evolves according to Darwinian principles of reproduction and selection of the fittest. That is, the individuals (solutions) that are better positioned in the search space according to the problem, maximization or minimization, have a high probability of surviving and reproducing in future generations. The reproduction is performed by variation operators (crossover and mutation mainly).

Since EAs are an emulation of the natural selection process, the following components are abstracted:

1. **Representation of solutions.** It links the real world with the EA. It can be at the phenotype (real solutions) or genotype level (bits or solutions represented by a code).
2. **Fitness quality function.** It represents the problem to be solved. Therefore, it defines the quality of the solutions, that is to say, it is the representation of the natural environment in which the individuals live and try to survive.
3. **Population of solutions.** It is a set of potential solutions to the problem.
4. **Parent selection mechanisms.** The role of this mechanism is to distinguish among individuals considering their quality, preferring, in principle, the best. Usually, it has probabilistic elements.
5. **Variation operators.** Their function is to create new individuals from existing ones.
6. **Replacement mechanism.** It aims to distinguish among individuals based on their quality to maintain a fixed population size; this mechanism is usually deterministic.

The main paradigms of the evolutionary algorithms are:

- Genetic Algorithms
- Genetic Programming
- Evolution Strategies
- Evolutionary Programming
- Differential Evolution

3.1.1 Genetic Algorithms

Genetic algorithms (GA) [45] are based on the mechanisms of natural selection and genetics. GAs combine the survival of the fittest among strand structures, with a structured but random information change, to form a search algorithm with some of the innovative talents of human search [45].

Algorithm 7 Simple Genetic Algorithm

```
1: Population start
2: while The stop criterion is not reached do
3:     Evaluate population
4:     Select Parents
5:     Apply recombination of parents
6:     Apply mutation of children
7:     Replace
8: end while
```

A GA may represent its solutions at a genotype or phenotype level, depending on the type of problem being addressed. The GAs contain the elements of the EAs: parent selection, probabilistic techniques (roulette, surplus and universal stochastic, deterministic sampling and tournaments), sexual recombination or crossing (uniform, complete and simple arithmetic), mutation (simple, uniform or rearrangement) and replacement. The general GA process is shown in Algorithm 7.

3.1.2 Genetic Programming

Genetic programming (GP) [46] operates similarly to genetic algorithms; the main difference is the representation of solutions. Whereas GAs individuals can be represented by means of vectors or bit strings, individuals in GP are represented by tree structures, which make GPs representation seen as non-linear structures.

The goal of GP is to let computers learning a problem-solver without being explicitly programmed, generating solutions to problems that come out of program induction. The programmer does not specify the size, shape, and structural complexity of the solution programs, but the programs evolve to generate satisfactory solutions [47].

Algorithm 8 General Genetic Programming

1: Randomly create an initial population of programs from the available primitives
2: **while** The stop criterion is not reached **do**
3: Evaluate population by executing each program
4: Select Parents
5: Apply recombination of parents
6: Apply mutation of children
7: Replace
8: **end while**

Algorithm 8 shows the GP process to evolve programs. In GP the variables and constants of the representation are called *terminals* (that is, the leaves of the tree) and the arithmetic operators of the representation are called *functions* (that is, the nodes of the tree). The set of terminals and functions together form the primitive set of a GP system.

3.1.3 Evolution Strategies

Evolution Strategies (ESs) [48] emulate evolution at individual level, then there is a crossover-operator, either sexual or panmictic (more than two parents); this operator is secondary in ESs [49]. The main operator for this type of algorithm is the mutation that uses random numbers generated by a Gaussian distribution. This strategy has the particularity that mutation values vary over time and are self-adaptive. The representation in ESs is at the phenotypic level, and the process of selection of survivors is deterministic and extintive, i.e., the worst individuals have zero probability of surviving.

Algorithm 9 Evolution Strategies

1: Generate randomly an initial population of solutions.
2: Calculate the fitness of the initial population.
3: **while** The stop condition is not reached **do**
4: Select two or more parents randomly.
5: Apply (optionally) crossover to generate offspring.
6: Apply the mutation operator to the offspring.
7: Evaluate offspring.
8: Select the best individuals for the next generation based on their aptitude.
9: **end while**

The selection of parents is not biased by the fitness values, i.e., they are randomly selected from the population with a uniform distribution. Each parent can be selected more than once or cannot be selected [49]. ES process is shown in Algorithm 9.

3.1.4 Evolutionary Programming

Evolutionary Programming (EP) [50] emulates evolution at the species level. Therefore, there is no cross-breeding operation, since in nature it is not possible to cross different species. The survivor selection technique is based on stochastic tournaments where parents and offspring compete and those who get higher wins in the tournament survive. The wins are determined by comparing the fitness of each individual with the fitness of other individuals of the population chosen at random.

Algorithm 10 Evolutionary Programming

1: Generate randomly an initial population of solutions.
2: Calculate the fitness of the initial population.
3: **while** The stop condition is not reached **do**
4: Apply the mutation to the entire population to obtain offspring.
5: Evaluate offspring.
6: Select (By Stochastic tournaments) individuals for the next generation.
7: **end while**

The mutation operator is the only variation operator. In EP there is no parent selection because each parent generates an offspring through the mutation operator (see Algorithm 10). For the stochastic tournament, it is essential to define the number of encounters for each individual in the population. After having performed all the matches, the whole population (offspring and parents) are ordered by the number of wins and not by the fitness value, to assure diversity in the evolutionary process. Finally, the first half of the entire population remains for the next generation.

3.1.5 Differential Evolution

Differential Evolution (DE), introduced by Storn and Price in [6], is based on the evolutionary principle. Differences between vectors (individuals of the population) are performed during the variation operator mechanism. The evolutionary process of the standard DE, also known as *DE/rand/1/bin*, consists of four main steps: (1) initialization, (2) mutation, (3) recombination, and (4) selection. While the first step is called once at the beginning of the process, the last three steps are repeated during the DE generations G until a termination criterion is reached.

In DE an optimization problem solution is known as a vector

$$X_{G,i} = (x_{G,i,j}, \ldots, x_{G,i,D})$$

where $X_{G,i}$ represents a vector i at generation G, and D is the number of decision variables (i.e. the search space dimensionality, $D = n$). In the same way, a population is represented as $P_G = (X_{G,i}, \ldots, X_{G,Pmax})$ where $Pmax$ is the fixed population size at each generation G. *MaxFEs* is the maximum number of fitness evaluations allowed for the algorithm, i.e., the termination criterion.

Step 1. Initialization. A set of user-defined number of $Pmax$ vectors $X_{G,i}$ are generated randomly. In this step, the dimension D and the upper (x_{up}) and lower (x_{lo}) boundaries are considered, as follows:

$$X_{G,i} = x_{lo} + rand \times (x_{up} - x_{lo}) \tag{3.1}$$

Step 2. Mutation. During this process, every vector $X_{G,i}$ within the population P_G is mutated by computing Equation (3.2):

$$v_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j}) \tag{3.2}$$

where $v_{G,i,j}$ is an element of the mutant vector, $\{r0 \neq r1 \neq r2\} \neq i$ are random selected vectors between 0 and $Pmax - 1$ within the population. F is the scaling factor, an user-defined parameter.

Step 3. Recombination. The mutant vector $V_{G,i}$ mixes its elements with the target vector $X_{G,i}$ using Equation 3.3 to generate a trial vector $U_{G,i}$.

$$u_{G,i,j} = \begin{cases} v_{G,i,j} \text{ if } rand_{i,j}[0,1] \leq Cr \text{ or } j = Jrand \\ x_{G,i,j} \text{ , otherwise} \end{cases} \tag{3.3}$$

where $rand_{i,j}[0,1]$ is a real random number, Cr is a user-defined parameter and $J_{rand} \in rand[1, D]$ is a random selected position of the vector.

Step 4. Selection. It determines whether the target $X_{G,i}$ or the trial vector $U_{G,i}$ survives to the next iteration $G + 1$

$$X_{G+1,i} = \begin{cases} U_{G,i} \text{ if } U_{G,i} \text{ is better or equal than } X_{G,i} \\ X_{G,i} \text{ , otherwise} \end{cases} \tag{3.4}$$

The complete process is described in Algorithm 11.

In the specialized literature, different DE variants have emerged to improve the mutation and crossover operators. The general convention used for naming the DE variants is **DE/a/b/c**, where **a** represents a string denoting the base vector, and **b** is the number of difference vectors considered for the perturbation of **a**. Finally, **c** depicts the type of crossover being used (*exp* and *bin* for exponential and binomial, respectively).

Four most frequently referred mutation strategies, in addition to **DE/rand/1** presented in Equation 3.2, are listed below:

DE/best/1:

$$v_{G,i,j} = x_{G,best,j} + F(x_{G,r0,j} - x_{G,r1,j}) \tag{3.5}$$

DE/current-to-best/1:

$$v_{G,i,j} = x_{G,i,j} + F(x_{G,best,j} - x_{G,i,j}) + F(x_{G,r0,j} - x_{G,r1,j}) \tag{3.6}$$

Algorithm 11 DE/rand/1/bin

Randomly generate an initial population of vectors $P_0 = (X_{0,i}, \ldots, X_{0,Pmax})$
Calculate the fitness of each vector in the initial population.
repeat
  for $i \leftarrow 1, Pmax$ **do**
    Randomly select $r0$,$r1$,$r2 \in [1, Pmax]$ and $r0 \neq r1 \neq r2 \neq i$
    Randomly select $J_{rand} \in [1, D]$
    for $j \leftarrow 1, D$ **do**
      if $rand_j \leq Cr$ Or $j = Jrand$ **then**
        $u_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j})$
      else
        $u_{G,i,j} = x_{G,i,j}$
      end if
    end for
    if $U_{G,i}$ is better or equal than $X_{G,i}$ **then**
      $X_{G+1,i} = U_{G,i}$
    else
      $X_{G+1,i} = X_{G,i}$
    end if
  end for
  $G = G + 1$
until Stop condition is reached

DE/best/2:

$$v_{G,i,j} = x_{G,best,j} + F(x_{G,r0,j} - x_{G,r1,j}) + F(x_{G,r2,j} - x_{G,r3,j}) \tag{3.7}$$

DE/rand/2:

$$v_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j}) + F(x_{G,r3,j} - x_{G,r4,j}) \tag{3.8}$$

The indices $r0$, $r1$, $r2$, $r3$ and $r4$ are mutually exclusive integers randomly chosen from the range [0, $Pmax - 1$], and all are different from the base index i. Whereas, $x_{G,best,j}$ is the best individual vector with the best fitness value in the population at generation G.

Regarding the recombination operator. In addition to the binomial crossover described in Equation 3.3 which is performed on each one of the D variables from the mutant vector $v_{G,i,j}$ having a (nearly) binomial distribution [7], there exists another crossover alternative for DE, named exponential crossover (see Algorithm 12). However, this operator is effective only when linkages exist between the neighboring decision variables [51].

Algorithm 12 Exponential crossover

$u_{G,i} = x_{G,i}$, j is randomly selected from $[1, D]$, $L = 1$;
repeat
  $u_{G,i,j} = v_{G,i,j}$, $j = (j + 1)$ modulo D
  $L = L + 1$
until $rand[0, 1] < Cr$ and $L < D$

3.2 Swarm Intelligence Algorithms

Unlike EAs, Swarm Intelligence Algorithms (SIAs) [52] are based on the behavior of animals working in groups, either to acquire food, to defend themselves against predators or to group for obtaining a benefit-common ground. SIAs differ from EAs in the way they search for the optimal solution in the search space.

The population in SIAs is characterized by collaboration and not by competition, as it is the case with EAs. In addition, there is usually no replacement, and the population remains constant during all iterations.

The main paradigms of Swarm Intelligence are:

- Particle Swarm Optimization.
- Ant Colony Optimization.
- Artificial Bee Colony.
- Bacterial Foraging Optimization

3.2.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is based on the simulation of the social behavior of birds within a flock [42]. Individuals are called particles in PSO and fly through the multidimensional search space. Particles movements are based on the socio-psychological tendency of individuals to emulate the success of other individuals.

Algorithm 13 Generic Particle Swarm Optimization

```
1: Randomly generate an initial set of particles.
2: Calculate the fitness of the initial swarm.
3: while The stop condition is not reached do
4:    Select the leader(s) of the swarm
5:    Update the update position (flight)
6:    Evaluate fitness function for each particle
7:    Update the particle memory
8: end while
```

Each particle represents a potential solution to the problem; its position changes according to its own experience and that of its neighbors, a set of such particles constitutes a swarm. Algorithm 13 shows the generic PSO process.

3.2.2 Ant Colony Optimization

Ant Colony Optimization (ACO) [53] is a meta-heuristic that includes a set of optimization techniques inspired by the collective behavior of foraging ants, which are able to find a short path between the nest and the food source by means of communication through traces of artificial pheromones.

Each agent (ant) in ACO leaves a pheromone trace in every movement within the search space (directed graph). The pheromone trace is more intense in areas of the search space where more ants have passed than other zones. To handle the pheromone trace in ACO a matrix of real numbers is used.

The success of ACO to find the shortest paths is because it will, over time, be more loaded with pheromones, while the longer path will not contain high concentrations of pheromones, due to the evaporation that occurs over time.

Algorithm 14 Ant Colony Optimization

```
1: while Stop condition is not reached do
2:    Construction of solutions for ants
3:    Update pheromone matrix
4:    Control of actions
5: end while
```

In ACO there are three fundamental procedures [53], see Algorithm 14:

- **Construction of solutions by ants.** It manages the ant colony which run through the search space (a directed graph) randomly. Each ant can be moved by applying stochastic decision making using information from pheromone traces and heuristic information. In this way, the ants construct a suboptimal solution to the problem.
- **Update pheromone.** Pheromone trace values can be increased because the ants deposit pheromone in each component or connections they use to move from one point to another in the search space (a graph in this case).
- **Control of actions.** A procedure used to carry out central actions that ants can not develop individually.

3.2.3 Artificial Bee Colony

Artificial Bee Colony Optimization (ABC) [54] is based on the foraging behavior of honey bees.

Algorithm 15 General Artificial Bee Colony Optimization

```
1: Initialize swarm of bees with food sources
2: Evaluate swarm of bees
3: while Stop condition is not reached do
4:    Apply phase of employed bees
5:    Apply phase of onlooker bees
6:    Apply phase of scout bees
7: end while
```

The main ABC feature (Algorithm 15) is based on the communication among bees by means of dances. Bees are operators and solutions are the food sources (flowers). Employed bees perform exploration, onlooker bees carry out exploitation and scout bees introduce diversity by adding random solutions [54].

3.2.4 Bacterial Foraging Optimization

Bacterial foraging optimization algorithm (BFOA) [55] is inspired by the social foraging behavior of Escherichia coli. BFOA considers the form of displacement, reproduction, and elimination-dispersion of bacteria.

On the other hand, a modified BFOA (MBFOA) [56] reduces the number of conditions required to execute the original BFOA (see Algorithm 16). In MBFOA the communication among bacteria is simplified, which produces a collaborative environment among them allowing to find areas with high contents of nutrients, that is to say, the best solutions of the problem.

Algorithm 16 Modified Bacterial foraging optimization algorithm

```
Generate randomly a initial swarm of bacteria
Evaluate swarm
while Stop condition is not reached do
    repeat
        Perform chemotactic process for each bacteria
    until Stop condition
    Perform reproduction process
    Eliminate the worst bacterias within the swarm
    Generate new bacteria randomly.
end while
```

3.3 Artificial Immune System

The human immune system inspires the artificial immune system (AIS) that uses associative learning, memory, and retrieval. AIS can be applied to solve optimization problems, pattern recognition, and task classification codes [57].

AIS emerged in the 1990s as a new branch in Computational Intelligence (CI) [58] and it is based on four theories of immunity: the classical theory, clonal selection, network-based and danger-based.

The classical theory is that the recognition of antigens (foreign elements in the body) motivates the generation of antibodies that destroy them. Epitopes (segments on the surface of antigens) and paratopes (antibody surface segments) serve to measure the affinity between them.

Algorithm 17 Artificial Immune System

```
Initialize a set of ALCs C
Determine set of antigens D_t
while Stop condition is not reached do
    for each antigen z_p in D_t do
        Select a subset of ALCs S to expose them to z_p
        for Each ALC x_i in S do
            Calculate the affinity with the antigen z_p
            Select a subset of S, called H with the ALCs with the highest affinities
            Adapt ALCs of H with some selection method based on affinity with the
            antigen and/or with affinity in the network among ALCs of H
            Update the stimulation level of each ALC in H
        end for
    end for
end while
```

The clonal selection mechanism establishes the idea that only those immune cells (B lymphocytes) that best react to the stimulation of an antigen will be cloned. Antigens are molecules that are expressed on the surface of pathogens that can be recognized by the immune system and that are also capable of initiating the immune response to remove them [59].

AIS, following network-based theory, is characterized by B-cells interconnecting to form networks. i.e. lymphocyte can be stimulated by another neighboring lymphocyte as well as by an antigen [57].

Finally, AISs that follow the theory based on danger are distinguished by the fact that immune cells, in addition to responding to antigens, are also able to react to cells that represent danger, including other agents of the same body.

Algorithm 17 shows the general process of an AIS.

3.4 Constraint-Handling for Evolutionary Algorithms

Because EAs, SIAs, and AISs were designed for unconstrained problems, it is necessary to incorporate particular mechanisms that allow them to deal with feasibility information in a constrained numerical optimization problems (CNOPs). Constraint-handling techniques include information either in the selection processes or the variation operators to guide the search [60]. In [34] several techniques can be found for constraint-handling. They are classified according to the type of integration with the objective function, see [34]. However, in this section, only two mechanisms used in the implementations of this research are described.

3.4.1 Feasibility Rules

It is one of the most popular techniques for handling constraints [61]. In this approach, a set of three feasibility criteria are used to compare a pair of solutions:

- When comparing two feasible solutions, the one with the best objective function is chosen.
- When comparing a feasible and an infeasible solution, the feasible one is chosen.
- When comparing two infeasible solutions, the one with the lowest constraint violation sum is chosen.

The constraint violation sum can be calculated as follows:

$$\phi(X) = \sum_{i=1}^{m} max(0, g_i(X))^2 + \sum_{j=1}^{p} |h_j(X)| \tag{3.9}$$

where the values of all inequality constraints $g_i(X)$, $i = 1, 2, \ldots, m$ and all equality constraints $h_i(X)$, $j = 1, 2, \ldots, p$ are normalized.

3.4.2 The ε-Constrained Method

The ε-constrained [62] method transforms any constrained numerical optimization problem into an unconstrained optimization problem, by using a tolerance (ε) which is descreased during the generations of the algorithm. The constraint violation ϕ of a given vector $X_{G,k}$ is computed as the sum of the amounts of all violated constraints, see Equation (3.10).

$$\phi(X_{G,k}) = \sum_{i=1}^{m} max(0, g_i(X_{G,k})) + \sum_{j=1}^{p} max(0, |h_j(X_{G,k})| - \delta) \tag{3.10}$$

The ε tolerance allows the so-called $\leq_\varepsilon$ comparison between two vectors by using only their fitness function values, see Equation 3.11.

$$X_{G,k} \leq_\varepsilon X_{G,l} \Leftrightarrow \begin{cases} f(X_{G,k}) \leq f(X_{G,l}) & \text{if } \phi(X_{G,k}), \phi(X_{G,l}) \leq \varepsilon \\ f(X_{G,k}) \leq f(X_{G,l}) & \text{if } \phi(X_{G,k}) = \phi(X_{G,l}) \\ \phi(X_{G,k}) \leq \phi(X_{G,l}) & \text{, otherwise} \end{cases} \tag{3.11}$$

The ε level is dynamically decreased at each generation as indicated in Equation 3.12:

$$\varepsilon(G) = \begin{cases} \varepsilon(0)(1 - \frac{G}{Gc})^{cp} & , 0 < G < Gc \\ 0 & , G \geq Gc \end{cases} \tag{3.12}$$

where cp is a user-defined parameter to control the reduction speed of the ε tolerance, G is the current generation number and Gc is the generation number when the ε value is set to zero. The initial ε value (i.e., $\varepsilon(0)$) is the constraint violation of the θth solution in the initial population, see Equation 3.13.

$$\varepsilon(0) = \phi(X_\theta) \tag{3.13}$$

Chapter 4

Memetic Differential Evolution for Constrained Problems: Syntactic Model and Taxonomy

Memetic computing (MC) is a topic that considers complex structures such as the combination of simple agents and memes, whose evolutionary synergies allow to solve complex problems. Hybridizations of two or more metaheuristics are part of MC subject. However, a particular class of optimization algorithms named memetic algorithms (MAs), whose structure is distinguished by an evolutionary framework and a set of local search components have been the base of MC.

In this chapter, the origins and the main MAs design principles are briefly described. Finally, an analysis of those memetic algorithms based on differential evolution for CNOPs is presented.

4.1 Baldwinian and Lamarckian Learning

There are two basic models of evolution that can be used to incorporate learning into a MA, the Baldwinian and Lamarckian Learning.

The Baldwinian learning allows an objective function value of a solution to be determined based on learning, i.e., the application of LSO. The result of the LSO does not change the genetic structure (genotype) of the solution, see Figure 4.1. On the other hand, the Lamarckian evolution, in addition to using learning to determine the objective function value of a solution, changes the genotype of a solution reflecting the result of the LSO.

Traditional schema theory does not support Lamarckian learning, i.e., making the genetic representation to match the solution found by the improvement procedure (local search operator). However, Lamarckian learning does alleviate the problem of multiple genotypes mapping to the same phenotype. While Baldwinian learning uses improvement procedures to change the fitness landscape, but the solution that is found is not encoded back into the genetic string [63].

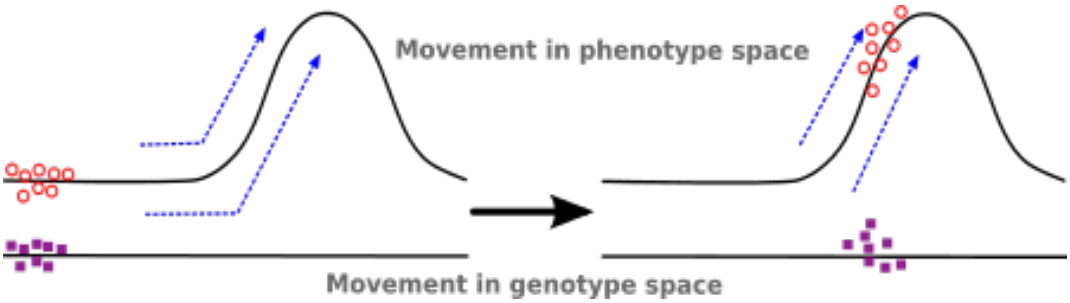


FIGURE 4.1: The Baldwinian learning implies that the movements of the phenotypes by learning entail, due to the cost, a movement and assimilation in the genotypes.

4.2 The Memetic Metaphor

Memetic algorithms (MAs) have been used over the years to improve the performance of several metaheuristics, either in convergence rate or accuracy in finding solutions to complex problems [64]. This term was introduced by Moscato in [9] to denote the interaction between a GA and a local search operator (LSO) to deal with the Traveling Salesman Problem (TSP).

The main idea of MAs is based on the the philosophical theory of Richard Dawkins (introduced in [65]). According to Dawkins human culture is composed by simple units named "memes". In this sense, a meme is the minimum unit of knowledge generated in human brains, and can be duplicated, modified, and combined with other memes to obtain a new meme. Memes are in cultural diffusion what genes are in the evolutionary process. However, they differ in that memes can be propagated vertically (in generations) and horizontal (during life-time learning), whereas the genes only propagate in vertical mode (inheritance). Likewise, the memetic approach coincides at a certain point with the Lamarckian model of evolution, which states that the environment gives rise to changes in animals and these changes are inherited to their descendants.

According to Neri and Cotta in [64] MAs can be defined as:

Memetic algorithms are population-based metaheuristics composed of an evolutionary framework and a set of local search algorithms which are activated within the generation cycle of the external framework.

On the other hand, Ong and Keane in [66] conceived the term "meta-Lamarckian learning" to introduce the idea of adaptively choosing multiple memes during a MA search in the spirit of Lamarckian learning.

Krasnogor et. al in [67] coined the term "multimeme algorithms" to refer to MAs that couple more than one meme.

Therefore, in this work, the term memetic algorithm is used for those that use a local search operator and multimeme algorithm for those that include multiple local search operators, to differentiate both schemes.

4.3 Syntactic Model of Memetic Algorithms

MAs were not proposed as specific optimization algorithms, but as a broad type of algorithms inspired by the diffusion of the ideas and composed of multiple existing operators. They allow the adaptation of particular operators in any bio-inspired algorithm that contribute to enhancing the performance and accuracy of the final results obtained. However, the broad scope of MAs could raise difficulties during their design process. In this way, Krasnogor and Smith [13] suggested to define a syntactic model in order to help the MA design.

The first issue to obtain a syntactic model refers to answering the question "What is the best tradeoff between local search and the global search provided by evolution?" [13], from which the following aspects are derived:

- The sequence of the local search procedure within the global search evolutionary cycle. In which part of the global search process the local search will be applied, i.e., before, during or after the variation operators.
- The frequency of local search activation. It refers to the number of times the local search operator will be activated. It can be determined in a probabilistic or deterministic way.
- The exploitation area of the local search. It refers to which individual(s) of the population will be the initial point of search for the improvement process.
- The computational effort allowed for the local search procedure. How much the local search operator will be permitted to exploit its neighborhood.
- The type of learning model (Lamarckian or Baldwinian)

These issues will be dictated by the type of problem to be solved, the features of the global search algorithm and the selected local search operator pivot rules (the way to generate neighbors during the exploitation process).

It is important to highlight that, as the number of local search operators to be used increases, the complexity of the syntactic model will increase. Therefore, the MA would require an extra mechanism to either, help coordinating the local search operators, or if applicable, have an accurate knowledge of the performance of each local search operator in the domain to which they will apply.

4.4 Memetic Differential Evolution for CNOPs

The integration of local search operators in evolutionary algorithms is a good alternative to improve results in different types of optimization problems including CNOPs. Therefore, in this section a set of DE-based MAs (MDE for short) for CNOPs are described.

According to the type of local search operator, the related approaches can be grouped into three sets: (1) direct-based operators, (2) gradient-based operators and (3) special operators.

4.4.1 MDEs with Direct-Based Operator

Muelas et al. in [68]. proposed a MA based on DE to solve continuous optimization problems. The authors used the multiple trajectory search algorithm as local search operator which was activated after a certain number of generations and was applied to the best solution of the population. Mandal et al. in [69] used an hybrid-mutation strategy to solve real-world optimization problems. The authors combined two techniques in the mutation process (DE/current-to-best/2 and DE/rand/1/bin). Moreover, the authors used the Solis and Wet's algorithm [70] as local search operator, which was applied for the best solution of the population, by using an activation frequency criterion. Hernández et al. in [71] proposed a MA based on DE as global search and a hill climbing method as local search operator to solve CNOPs. After the selection process in DE, the local search was activated for the best solution of the population. In a recent proposal [72] the authors improved the algorithm by combining two mutation operators. Domínguez et al. in [73] proposed a MA to solve CNOPs, the algorithm combined DE as global search and Powell's conjugate direction method as local search operator. The local search was activated during the main cycle of DE and was applied to the best, the worst and a random selected solution. Zhang et al. in [74] proposed a MA that used DE and Hooke-Jeeves as local search operator to solve continuous optimization problems. The local search was activated by means of a probability for each solutions of the population during the DE process. Piotrowski in [75] combined DE and a local search operator inspired by the Nelder-Mead method to solve continuous optimization problems. The local search operator was activated by a probability and was applied to the best solution of the population during the DE generations. Vakil et al. in [76] proposed a memetic DE with a differential-bidirectional-random-search method as local search operator which was applied to all solutions during a user-defined number of generations of the global search process. The algorithm was tested in continuous optimization problems.

4.4.2 MDEs with Gradient-Based Operator

Takahama and Sakai in [15] used DE with exponential crossover (DE/rand/1/exp) and the ε-constrained method to solve CNOPs. They also adapted a gradient-based mutation as local search mechanism, which was activated for the solutions after applying the crossover operator. The authors presented an improved version based on a new control for the ε-tolerance. In a recent work [14], they proposed two novel mechanisms to control boundary constraints to further improve their approach.

4.4.3 MDEs with Special Operator

Liu et al. in [77] proposed a co-evolution-memetic-based algorithm to solve CNOPs. They used two different populations within DE. One population aimed to minimize the objective function regardless of constraints, and the other one ensured to minimize the constraints violation regardless of the objective function. The authors used a Gaussian mutation originally adopted in real coded genetic algorithms (GAs) as local-search-like operator, which was applied to the population when the best solution kept unchanged for several generations. Menchaca and Coello in [78] proposed a MDE algorithm to solve CNOPs which combined DE with a Nelder-Mead based

operator (called simplex operator). Using a frequency, the simplex operator was activated to generate new solutions of the population. Zhao et al. in [79] proposed a MA based on DE as global search and Cauchy distribution used as local search operator which was applied to the best solution at each generation. The algorithm was designed to solve continuous optimization problems. Pescador and Coello in [80] implemented a crossover-memetic-based algorithm to solve CNOPs. They proposed a DE with simplex crossover as local search mechanism. The authors applied the local search on the neighborhood of the best and the worst solutions of the population and was activated at each generation of the DE algorithm. Pan et al. in [81] proposed a DE algorithm with a local search operator based on Cauchy mutation to solve large-scale optimization problems. The local search operator was activated inside the DE generations when the global best solution did not improve on a certain number of generations.

From the works previously discussed, stand out the main aspects of algorithmic coordination, that are part of the syntactic model of MDE, such as the exploitation zone and activation frequency. However, there is no a specific design pattern, i.e., in some works a deterministic activation frequency was used, and the neighborhoods of all the individuals of the population were exploited [15, 14, 75], while in other approaches the neighborhood of the best individual of the population was exploited [68, 71, 72]. On the other hand, some probabilistic schemes used a user-defined parameter to modulate the frequency of local search application [74], and others used information from the population to apply or not the local search operator [81, 77].

The Lamarckian learning model is present in all the MDE mentioned above. Moreover, in most approaches, the exploitation procedure occurs after applying the DE variation operators, except in [78] where the local search was implemented as a variation operator.

4.5 Multimeme Differential Evolution

Although there are multimeme syntactic models based on DE (MmDE), there is not one that addresses constrained optimization problems. However, in this section MmDE designed for unconstrained problems are described, which can be classified, according to the local search coordination mechanism, into three groups: (1) MmDE based on fitness diversity, (2) MmDE based on random selection, and (3) MmDE based on meta-Lamarckian learning.

4.5.1 Multimeme DE Based on Fitness Diversity

Neri et al. in [82] proposed an adaptive multimeme approach named AMmA, which works with three LSOs: localized random search (LRS), steepest descent explorer (SDE) and simulated annealing (SA). The local search coordination was based in a population diversity index designed for flat fitness landscapes, see Equation 5.1:

$$\psi = 1 - \left| \frac{f_{avg} - f_{best}}{f_{worst} - f_{best}} \right| \tag{4.1}$$

where f_{worst} , f_{best}, and f_{avg} are the worst, best, and average fitness function values in the population, respectively. Depending on the population diversity index value ψ, the LSOs were systematically applied as follows: LRS exploited the search space area of a randomly selected solution, SDE applied to the best solution and finally, SA applied on the second best solution. In some cases, SDE was used to exploit the solutions obtained by LRS and SA.

In other work, an adaptive evolutionary algorithm with intelligent mutation (IMLS) was introduced by Neri et al. [83], which coordinated three local search operators: increasing diversity (ID), greedy descent (GD) and steepest descent (SD). A diversity coeficient was used to activate the IMLS, see Equation 4.2:

$$\xi = min \left\{ \left| \frac{f_{best} - f_{avg}}{f_{best}} \right| , 1 \right\} \tag{4.2}$$

where f_{best}, and f_{avg} are the best, and average fitness function values in the population, respectively. In addition, ID and GD were applied on a randomly selected solution, while SD was used to exploit the neighborhood of the best solution.

Tirronen et al. in [84] proposed a MmDE to design a Finite Impulse Response filter for detecting weak defects in paper production. The Stochastic Local Searcher (SLS) and Hooke-Jeeves algorithm (HJ) were coordinated by a measurement of the fitness diversity and distribution of the fitness values within the population, see Equation 4.3:

$$\nu = min \left\{ 1, \frac{\sigma f}{|f_{avg}|} \right\} \tag{4.3}$$

where $|f_{avg}|$ and σf are, respectively, the average value and standard deviation over the fitness values of solutions in the population. ν was updated every 1000 fitness evaluations. For ν values smaller than 0.5, SLS and HJ were activated systematically. Subsequently, in [26] a multimeme approach was proposed, namely Enhanced Memetic Differential Evolution (EMDE), where three local searchers were coordinated: simulated annealing (SA), stochastic local search (SLS) and the Hooke-Jeeves algorithm (HJ). A scheme for the local search coordination based on exponential probability distribution was implemented. A fitness diversity measurement (see, Equation 4.3) and distribution of the fitness values within the population were used to activate the local searchers. SA was used to exploit the neighborhood of a randomly selected solution, and HJ was applied to the best solution. On the other hand, SLS was used in both cases: the best and a randomly selected solution.

Caponio et al. in [28] proposed a MmDE named super-fit memetic differential evolution (SFMDE), where a Particle Swarm Optimization (PSO) algorithm was applied in the beginning of the optimization process by helping to generate a super-fit solution. The Nelder-Mead method (NM) and the Rosenbrock algorithm (RA) were coordinated by using a probabilistic scheme. The activation mechanism was in function of a quality index of the super-fit solution, which was measured by Equation 4.4:

$$\chi = \frac{|f_{best} - f_{avg}|}{max|f_{best} - f_{avg}|_k} \tag{4.4}$$

where f_{best} and f_{avg} are the fitness values of the best and average solutions of the population, respectively; $max|f_{best} - f_{avg}|_k$ is the maximum difference observed (e.g., at generation k), beginning from the start of the optimization process. NM was applied to a randomly-selected solution, while (RA) was applied to the best solution.

The probability of local search activation is handled by Equation 4.5 which is based on the Beta distribution $B(\alpha, \beta)$

$$P = \frac{1}{B(\alpha, \beta)} \cdot \frac{(\chi - a)^{(\alpha-1)}(b - \chi)^{(\beta-1)}}{(b - a)^{(\alpha+\beta-1)}} \tag{4.5}$$

where a and b are, respectively, the inferior and superior limits of the distribution, and χ is calculated by Equation 4.4.

4.5.2 Multimeme DE Based on Random Selection

Iacca et al. in [29] presented a MmDE that consisted on an ensemble of parameters and strategies of DE empowered by a pool of local search operators (EPSDE-LS) for unconstrained optimization problems. EPSDE-LS coordinated three LSOs: Nelder-Mead, Powell conjugate direction method, and Rosenbrock algorithm. LSOs were controlled by means of an user-defined parameter, named local search activation frequency. On the other hand, EPSDE-LS applied a random-based selection mechanism to activate one of the three LSO. The LSO selected exploited the best solution of the population.

4.5.3 Multimeme DE Based on Meta-Lamarckian Learning

Sabar et al. in [30], introduced a heterogeneous cooperative co-evolution MmDE (CC-HDE) to solve big data optimization problems. The algorithm coordinated two LSOs: the Rosenbrock algorithm (RA) and the Powell's conjugate direction method (PCD). LSOs were controlled by a selection mechanism based on a reward-punishment scheme to evaluate the effectiveness of the applied LSO. The selection method was assisted by the Page–Hinkley (PH) statistical test to decide if the current LSO is not performing well anymore and a new one should be selected. The selection mechanism is also used to control other elements in the algorithm, such as operators and parameter values. The reward-punishment scheme measured the LSO performance as follows:

$$r_{i(G)} = r_{i(G)} + \Delta \tag{4.6}$$

and

$$r_{j(G)} = r_{j(G)} - \frac{\Delta}{NLSO - 1}, \forall j \in \{1, 2, \ldots, NLSO\} \text{ and } i \neq j \tag{4.7}$$

$$\Delta = \frac{f_{before} - f_{after}}{f_{before} + f_{after}} \tag{4.8}$$

where f_{before} and f_{after} are the fitness value of the solution before and after the application of the local search operator, respectively.

If i^{th} LSO cannot improve the solution:

$$r_{i(G)} = r_{i(G)} - |\Delta * IG| \tag{4.9}$$

and

$$r_{j(G)} = r_{j(G)} - \frac{|\Delta| * IG}{NLSO - 1}, \forall j \in \{1, 2, \ldots, NLSO\} \text{ and } i \neq j \tag{4.10}$$

where $r_{(G)}$ is the impact of i^{th} LSO at generation G, IG is the current generation divided by the total number of generations, and $NLSO$ is the number of LSOs.

4.6 Multimeme Evolutionary Approach for CNOPs

This section describes a multimeme model based on agents designed to solve CNOPs. Although it is not a MmDE, it is important to note that the coordination of local search operators in CNOPs is based on Lamarckian learning. While constraints are handled with the feasibility rules, see Section 3.4.1.

4.6.1 Agent-based Memetic Approach

Ullah et al. [31, 32] proposed an agent-based memetic algorithm (AMA) to solve CNOPs, which coordinated four LSOs adaptively by means the local search performance during an evolutionary approach. The first and second local searchers were random-based implementations, while the third operator was a gradient-based algorithm. Finally, the fourth LSO was a simplified direct method. Each LSO was randomly applied on a solution, but with the passing of generations, the LSOs with better improvement index were selected in order to improve the exploitation of the search space. The local search performance measure proposed is shown in Equations 4.11 and 4.12, and is described as follows: When the LSO starts with an infeasible solution and becomes feasible after the application, *I.I.* gets the value 1. If the situation is the opposite then *I.I.* is assigned -1. However, if the vector remains feasible before and after the local search operator, *I.I.* is calculated by Equation 4.11:

$$I.I. = \frac{f_{before} - f_{after}}{f_{before}} \tag{4.11}$$

where f_{before} and f_{after} are the fitness value of the solution before and after the application of the local search operator, respectively. In the same way, if the local search operator starts with an infeasible vector and still results an infeasible, *I.I.* is obtained by Equation 4.12:

$$I.I. = \frac{\phi_{before} - \phi_{after}}{\phi_{before}} \tag{4.12}$$

where ϕ_{before} and ϕ_{after} are the constraint violation sum of the solution before and after the application of the local search operator, respectively. An *I.I.* value is restricted between -1 and 1. The values outside these ranges are bounded to the permitted limits.

4.7 Memetic Differential Evolution for Real World Applications

This section describes memetic approaches based on DE that address real-world problems. This revision includes some hybrid algorithms based on DE (HDE for short) which, although they are not as such, memetic algorithms, are part of the memetic computation.

Fan and Lampinen in [85] proposed a Trigonometric Mutation Operation to DE (TDE) for continuous optimization problems. The algorithm alternates, by a probabilistic mechanism, the mutation operator of rand/1/bin and the proposed trigonometric mutation method, which works as a local search operator. The probability value used to perform the local search was rather low (0.05). The algorithm was tested in an aerodynamic five-hole probe calibration problem that consists of training an artificial neural network. Despite the TDE higher convergence velocity regarding the standard DE and the Back-Propagation algorithms, the addition of an user-defined parameter, to control the probability of the trigonometric mutation activation, increases the parameter tuning process.

On the other hand, Hu et al. in [86] applied TDE to reduce the impact of surplus harmonics in programmed pulse-width modulation (PWM) by pushing the first crest of the surplus harmonics backwards, ameliorating the amplitude frequency spectrum distribution of the output waveform. In this work, the probability to perform the local search was reduced to 0.02 to avoid premature convergence and to guarantee low extra computational cost.

Santamaría et al. in [87], proposed three MDEs approaches for 3D reconstruction of forensic objects through range image registration. Every MDE were based on DE/rand/1/bin, and coordinates one local search method (Powell's conjugate direction method [88], Solis & Wets' method [89], and Crossover-based Local Search method [49], respectively). The global-local search coordination was performed on a probabilistic approach, i.e., it was based on a random application with uniform distribution considering a probability value of 0.0625. Every MDE applied the local search method on the best individual of the population.

Leskinen et al. in [90] studied the performance of two MDE approaches on the Electrical Impedance Tomography (EIT) problem. Both MDEs were based on a self-adaptive DE scheme. The first MDE named Variation Operator Local Search Differential Evolution (VOLSDE) used the Sequential Quadratic Programming (SQP) on the scale factor during the generation of the offspring. The second MDE, named Lifetime Learning Local Search Differential Evolution (LLLSDE) performs the Nelder-Mead method (NM) on a pseudo-randomly selected individual of the population. A probabilistic mechanism was used to activate local searchers. VOLSDE activates the SQP for 20 fitness evaluations of the global search and uses 0.1 as a probability value. LLLSDE applied the NM for 50 fitness evaluations using 0.2 as a probability value.

Fu and Yu in [91] developed a MDE and applied it to train an artificial neural network to construct a practical soft-sensor of jet fuel endpoint of the main fractionator of hydrocracking unit. The memetic approach, called HEODE, combined the DE/rand/1/bin and the Extremal Optimization (EO) with Adaptive levy mutation as a local search operator. HEODE implemented a probabilistic mechanism to activate the local search operator during the selection process. Authors defined a low probability (0.1) to enable the EO.

Cruz-Ramírez et al. in [92], implemented a Memetic Pareto DE multiobjective evolutionary algorithm, to learn the structure and weights of the Neural Networks for designing optimal artificial neural network models with sigmoid basis units for multiclassification tasks in predictive microbiology. The proposed approach used as local search the improved Resilient Backpropagation with backtracking-$iRprop^{+}$. The local search activation was configured in a deterministic form since it was applied

in three generations during the DE evolutionary cycle. The algorithm considered conditions in the first Pareto front to select the individuals to use the local search.

Neri and Mininno in [93] designed a Memetic compact DE. The optimization problem tackled was the training of a Recurrent Neural Network, which is used to control a Cartesian robot control system. A Stochastic Local Search was implemented in a probabilistic way, since a probability value was considered to activate the local search on the best individual in the population. For the experiments reported, the probability value to activate the local searcher was configured with a small value (0.005) in order to reduce the activation possibility

Li and Yin in [94] optimized the design of a reconfigurable antenna array with discrete phase shifters by means of a memetic approach based on DE/rand/1/bin and the Artificial Bee Colony (ABC) [95]. The ABC was adapted as a local search operator, which is activated during the DE mutation process by a probabilistic mechanism. The probability value performed by the algorithm was 0.2, to take advantage of the exploration capability of DE and the stochastic exploitation of the ABC. Therefore, DE/ABC can overcome the lack of the exploitation of the DE algorithm. In other work [11], the authors implemented a DE/ABC variant to optimize the parameter estimation for chaotic systems. Unlike the DE/ABC version, the DE/rand/2/bin mechanism was adapted in the approach.

Basetti and Chandel in [96] developed a MDE by combining the DE/rand/1/bin and the Taguchi method [97] to optimize a hybrid power system state estimation. The algorithm proposed used the DE/rand/1/bin coupled with the Taguchi method as local search operator in a deterministic way, between DE's crossover and selection operator. The Taguchi method exploited the search space of the entire trial population.

Elsayed and Sarker in [98] proposed a MDE based on a multi-operator scheme to solve a big data optimization problem. The algorithm combines three mutation operators with the binomial crossover. Besides, the interior point method, which uses Newton's method to handle the equality constraints, was implemented as local search within the MDE, The coordination between global and local search was defined in a deterministic way since it is activated every generation on the best individual of the population during the global evolution process.

Yi et al. in [22] devolped a εDE with a local search based on a mutation operator (εDE-LS) to solve constrained numerical optimization problems. εDE-LS integrated a special local search movements by deterministic rule, that considers the number of feasible solutions. The local search is based on DE/current-to-rand/2 mutation, but instead of select vectors randomly, the authors consider feasible solutions. The approach was tested in well-known benchmark problems, and a engineering design problem (car side impact design) which consists on minimizing the total weigh of the side-impact element of a car.

Surender Reddy et al. in [99] solved the generation scheduling approach for a microgrid comprised of conventional generators, wind energy generators, solar photovoltaic (PV) systems, battery storage, and electric vehicles. The proposed approach combines the standard DE and the Harmony Search algorithm (DE-HS). DE-HS applies the HS as local search mechanism just after DE selection process in a deterministic way. Due to HS is a multi-population metaheuristic, the whole population of DE is used by HS as initial population. After the local HS procedure, the worst individual in the DE population is replaced by the best individual obtained by HS.

Ali et al. in [100] proposed a MDE with the Simulated Annealing (SA) as local search operator for minimizing the molecular potential energy function. The proposed approach includes the SA after the DE/rand/1/bin evolutionary cycle in a deterministic way. SA exploit the best individual of the population. According to the authors, SA can help the proposed algorithm to escape from trapping in local minima and increases the diversity of the search.

Khan et al. in [101] performed a MDE to align map images to 2D GIS models. The memetic approach combines an adaptive version of differential evolution called jDE [102] (DE/rand/1/exp), and the S3 algorithm proposed in [103], which is a steepest descent local search operator. The memetic jDE exploited each trial solution of the population in a deterministic way by performing the S3 algorithm.

Ma et al. in [104] solved a dynamic scheduling problem in robotic cells by means of a memetic approach based on a discrete differential evolution (DDE) [105] and forward-backward earliest starting time algorithm (FBEST) [106]. The authors modified the standard FBEST to generate local optimal solutions. DDE implements the standard mutation and crossover operators of DE, i.e., DE/rand/1/bin. The extended FBEST was coupled in a deterministic way, and it was applied before the selection operator over the entire trial population.

Rakshit et al. in [107] implemented an adaptive MDE using Q-learning to solve a real-time multirobot path-planning problem. The proposed algorithm used the DE/current-to-best/1 variant and has two parameters called scaling factors, which are adaptively selected from a meme pool. According to the authors, an individual with a good fitness should search in the local neighborhood, whereas a poor performing individual should participate in the global search. The algorithm coordinates the meme pool by using the temporal difference Q-learning (TDQL), which works on the principle of reward and penalty and employs a Q-table to store the reward/penalty given to an individual of the population. The coordination of memes was performed by the Roulette choice function based hyper-heuristic scheme to adaptively select memes (scaling factors) for the individuals before participation in the DE.

Zhong et al. in [108] optimized a circuit tolerance design problem by means DE/rand/1/bin and a hybrid analysis method. The memetic approach implemented two approximation analysis methods of a circuit, the Vertex, and the Monte Carlo analysis. Whereas the first one has the advantage of low computational cost but can not ensure the accuracy of the solutions, the second method can obtain high accuracy solutions at the cost of expensive computation. Both methods were coordinated iteratively by an adjusting approximating rate which provides a flexible control on accuracy and time efficiency. When the measure gets the zero value, the Monte Carlo method was activated. On the other hand, the Vertex method was used in the evolutionary cycle. This mechanism was activated during the fitness evaluation process.

Tirronen et al. in [84] proposed a MmDE for designing digital filters which aim at detecting defects of the paper produced during an industrial process. The approach coordinates two local search algorithms: Hooke-Jeeves (HJ) and Stochastic Local Search (SLS). These local searchers are coordinated by means of an adaptive rule which estimates fitness diversity among individuals of the population. However, based on a prior study of the local search algorithms, different conditions were proposed to activate each one, i.e., if the fitness diversity value is between (0.25 and

0.4) SLS is applied to an individual pseudo-randomly chosen, while HJ is applied on the best individual in the population if the diversity value is less than 0.25.

Likewise, Tirronen et. al in [26] tackled the same problem of filter design for defect detection in paper production, but a different local search coordination mechanism was proposed. For this approach, Hooke-Jeeves (HJ), a Stochastic Local Search (SLS) and Simmulated Annealing (SA) algorithms were used as local search operators (LSO). The coordination mechanism was based on a probabilistic rule, which depends of fitness diversity estimation. Unlike [84], the conditions for each LSO were performed considering an exponential function, in order to increase or decrease the probabilities of using each LSO.

In [109] Caponio et al., optimized the design of a permanent magnet synchronous motor problem by means of a Multimeme Differential Evolution approach, which coordinates two LSOs, Nelder Mead (NM) and Rosenbrock algorithm (RA). The coordination mechanism considers the fitness value of the best individual of the population, called super-fit, to estimate how much the super-fit individual outperforms the remaining part of the population. The super-fit individual is generated by using a Particle Swarm Optimization (PSO) before starting the DE search process. In order to distribute the LSOs activation, a probabilistic mechanism based on a beta distribution was proposed. The parameters of the beta function were configured by means of a prior study of LSOs. NM is applied on eleven pseudo-randomly selected individuals, while RA is applied on the best individual in the population.

Chen and Wang in [110] tackled the crisp and fuzzy optimization problem for designing an optimal temperature control policy for a batch process of simultaneous saccharification and co-fermentation (SSCF) to produce ethanol from lignocellulose using the enzymes and the recombinant strain Saccharomyces yeast 1400 (pLNH33). The authors proposed a MmDE based on DE/rand/1/bin, which includes two operators (Acceleration and Migration) after the DE selection process. Whereas the acceleration operator was applied to speed up the convergent rate of the algorithm, the migration operator was activated to escape of local optimal solutions. Both operators were coordinated by a conditional status based on the population diversity. The synergy between both operators allowed the algorithm to use a small population size to achieve a global optimal solution.

Das et al. in [111] solved the spread spectrum radar poly-phase code design problem by a HDE and Simmulated Annealing method (AnDE). AnDE performs a variant of the DE/best/1/bin, which replaces the mutation operator with a center-mass-based mutation scheme. On the other hand, AnDE modifies the selection process by using a decreasing probability rule inspired by the Simmulated Annealing method.

Chung et al. in [112] proposed a HDE that combines the standard DE with Evolutionary Programming (EP) [113] to optimize a reactive power flow for economic and secure operation of power systems. The proposed approach, named DEEP, replaces the original mutation and crossover operators of DE by the EP mutation operator. According to the authors, this hybridization is effective in overcoming the disadvantage of DE that requires relatively large populations to avoid premature convergence.

Semnani et al. in [114] developed a HDE for solving two-dimensional inverse scattering problems. The hybrid method included the DE/rand/1/bin process and the truncated cosine Fourier expansion (TCFE) which is a classical method to tackle this

type of problems. The hybrid method considered two steps to integrate both algorithms. In the first step, a coarse scattered profile is obtained by using the TCFE to describe the scattered properties. The coefficients of the TCFE are iteratively estimated by means of DE. During the second step, the pulse function expansion (PFE) is adopted to improve the resolution of the profile. Subsequently, DE is applied to estimate the coefficients of the PFE. The hybrid method benefits from the advantages of both, TCFE and PFE representations of the scatterer properties.

Duvvuru and Swarup in [115] solved three cases of study of the economic load dispatch problem with valve point effect by a HDE approach. The proposed hybrid algorithm merges the interior point method (IPM) and DE/rand/1/bin. The HDE involved two steps. The first step uses IPM to minimize the cost function without considering the valve point effect. The second step considers value point effect and minimized the cost function using DE. On the other hand, Parassuram et al. in [116] tackled the same problem by means of a hybrid algorithm based on Particle Swarm Optimization (PSO) and DE/rand/1/bin. In this approach, during the evolutionary cycle, the population was divided in two sub-populations and each one was computed by DE operators and PSO operators, respectively. Subsequently, both sub-populations were merged and considered for next generations until the stop criterion was reached.

Fu et al. in [117] optimized a route planning for Unmanned Aerial Vehicle (UAV) on the sea by a hybrid approach composed by DE and a variant of Particle Swam Optimization (PSO) [118]. The hybrid approach is composed of two phases. The first phase implemented the quantum-behaved particle swarm optimization (QPSO) [119], while the second phase adapted the DE operators within the QPSO scheme. For this approach, the authors implemented a particular mutation operation that considers the best positions rather than the individuals of the population.

Jena et al. in [120] proposed a Differential Evolution which performs a hybridization of the mutation operator by means of a Gaussian Mutation (originally used in genetic algorithms) to optimize the combined heat and power economic dispatch problem. The proposed hybrid DE (HDE) were tested in three systems in order to measure the algorithm performance. The Gaussian Mutation is used during the complete search process, since it replaces the original DE mutation operator.

Zaman et al. in [121] developed a hybrid approach based on DE/rand/1/bin and the Interior Point Algorithm (IPA) [122] to 3-D Near Field Source localization, which is applied in Radar, Sonar, and digital communication. The proposed approach, called DE-IPA, implemented IPA following the finalization of the DE evolutionary cycle, in order to refine the best solution found.

Ganesan et al. in [123] solved the industrial green sand mould development problem, which is a multi-objective problem, by means a Game-theoretic DE (GTDE). The algorithm incorporates two deterministic strategies, based on the game theory, within the evolutionary cycle of DE. The first strategy modifies the influence of the trial vector on the principal parent at each generation, while the second strategy increases or reduces the mutation factor of the trial vector. Both strategies, namely defective and cooperative, respectively, are applied before executing the mutation operator and also are based on a decision rule.

Wang et al. in [124] tackled a nonlinear constrained multi-objective optimization problem of a Parallel Ankle Rehabilitation Robot by using a DE with an elitist-$(\mu+\lambda)$-selection process and a tournament selection mechanism. The main feature of the

selection process is the establishment of the new population which contains dominated and incomparable solutions after the one-to-one survivor selection. On the other hand, the authors modified the DE/best/1/bin mutation operator in order to consider some of the unfeasible solutions as the base vectors at the early generations and promote diversity. In addition, a second mutation strategy is considered to enhance the possibility of finding the global optimum.

Ma et al. in [125] designed a hybrid particle swarm optimization and DE (HHPSODE) and applied to pricing and lot-sizing decisions, which is a bi-level programming problem. HHPSODE restricted the particle's movement to the feasible region of the problem by modifying the velocity and position modulation. On the other hand, HHPSODE implemented a DE/rand-to-best/2/bin as sub-program to solve a lower-level problem. Finally, the PSO is used to solve the upper-level problem.

Dhaliwal and Dhillon in [126] integrated a Cat Swarm Optimization (CSO) [127] and DE for optimal infinite impulse response (IIR) filter design. The approach uses DE/rand/1/bin after the CSO evolutionary process to refine the search; the best individual obtained by DE is taken to replace the best individual of the CSO population.

Guo et al. in [128] proposed an enhanced self-adaptive Differential Evolution (ESADE) based on simulated annealing (SA) for rule extraction for recognizing oil reservoir. ESADE adapts the SA into the selection operator, and the control parameters with better performance were used in the next generation as initial control parameters. During the evolutionary cycle, control parameters of each target individual can be gradually self-adapted from their previous experience in getting promising solutions.

4.8 Final Remarks

Derived from the previous revision of the different memetic approaches based on DE, it can be highlighted that:

- No single recurring syntactic MDE model tackled CNOPs, i.e., different ways to global-local search coordination have proven to be efficient for MDEs in constrained search spaces.
- Regarding MDE and MmDE for real-world problems, there is a trend in the use of improved meme schemes with parameter adaptation mechanisms and the use of multiple mutation operators.
- The probabilistic approaches for the algorithmic coordination are a constant in the multimeme schemes.
- The Lamarckian learning model is present in all the memetic approaches described above.
- Most of the MDE and MmDE approaches implement zero-order local search operators, i.e., methods that do not require gradient function information.
- Multimeme schemes based on DE have not been studied for CNOPs.

The facts mentioned above are the motivation for this work since the implementation of multimeme approaches based on DE has not been addressed for constrained

optimization problems. Likewise, the complexity in the implementation of existing memetic approaches limits their usage to solve real-world problems. Therefore, that motivates us to carry out to design a simple multimeme coordination approach, which is introduced in this research.

It is important to mention that the most of the existing MmDE, which are based on the diversity of the population, require an a-priori study of local search operators to define the activation thresholds for local search operators during the DE evolutionary cycle.

On the other hand, most of the memetic approaches for solving CNOPs, described in this section, do not mention the influence that local search operators have on the final results. This might be because, in addition to local search, some approaches combine mechanism of parameter adaptation or combination of multiple operators. Therefore, the real benefit of using local search operators during the DE optimization process may not be discussed when solving CNOPs. These issues are addressed in the next chapter.

Chapter 5

Local Search Operators Influence in Memetic DE Approach for CNOPs

In this chapter, three studies on the influence of zero-order local search operators on memetic approaches are presented. Although for each of the studies, three different MDEs are used, all of them work under the same syntactic model, which consists of activating the local search operator using a probabilistic model.

5.1 Study 1: Local Search Performance Influence

This study is focused on measuring the performance of three direct local search operators (Nelder-Mead, Hooke-Jeeves and Hill Climbing, presented in Section 2.4) added separately into Differential Evolution, with the aim to relate it to the final results obtained by each MA variant in a constrained search space. The goal is to get a better understanding of the type of performance required by a local search operator in presence of constraints.

5.1.1 Performance Measure

The improvement Index (*I.I.*) proposed by Barkat et al. in [129] is adapted for constrained search spaces in this work to measure the local search performance. *I.I.* indicates the rate of fitness improvement made by a particular local search operator, and is calculated by Equations 4.11 and 4.12, presented in Section 4.6.1.

5.1.2 Algorithmic Coordination

The interaction between global and local search in this research work is the same for each approach compared. Three issues were considered in the global-local search coordination: (1) exploitation area, (2) application frequency and (3) type of replacement.

Exploitation Area According to Mezura-Montes et al. in [130] the DE variant used in this work (DE/rand/1/bin) has a good performance regarding exploration capabilities in constrained spaces. Due to that, in this proposal the best solution in the population was used to exploit promising areas in the search space by applying it the local search operator while using DE/rand/1/bin as the global search algorithm.

Application Frequency Inspired in [81, 77], the local search operator is activated if the best solution in the population does not improve after T generations. Besides, the algorithm considers a probability ψ given by Equation 5.1 to activate the local search operator. Although ψ was designed for a particular combinatorial optimization problem [82], in previous experiments it worked well in CNOPs.

$$\psi = 1 - \left| \frac{J_{avg} - J_{best}}{J_{worst} - J_{best}} \right| \tag{5.1}$$

where J_{best}, J_{worst} and J_{avg} are the best, worst and average of the fitness function values in the population, respectively. According to Neri et al. in [131] ψ is a population diversity index which measures it in terms of fitness. The population has high diversity when $\psi \approx 1$ and low diversity when $\psi \approx 0$. In this work, a higher diversity index value means a higher probability to apply local search.

Type of Replacement In this work, Lamarckian learning is used, i.e., the solution generated by the local search operator (X_{new}) is always kept for the next generation. The algorithm randomly selects a solution of the population (except the solution with the best fitness value) to be replaced by X_{new}. This is because it could be the case that X_{new} is worse than the best solution in the current population.

5.1.3 Memetic DE Approach

The MDE used in this study is shown in Algorithm 18, where the gray lines mark the elements to integrate the local search operator within DE/rand/1/bin.

5.1.4 Tolerance Value for Equality Constraints

For this study, the equality constraints are transformed into inequality constraints using a δ tolerance which decreases at each iteration as indicated in Equation 5.2 until a suitable value is reached.

$$\delta(G+1) = \frac{\delta(G)}{dec} \tag{5.2}$$

where dec is a user-defined parameter which determines how fast the δ value decreases over time and $\delta(0)$ is also a user-defined parameter.

5.1.5 Experiments and Results

The experiments are divided in two phases: (1) The proposed MDE+NM, MDE+HJ, and MDE+HC are tested on 18 benchmark problems, with different search space dimensionality (10 and 30 dimensions), used in the special session on "Single Objective Constrained Real-Parameter Optimization" in CEC'2010 [132], to analyze the performance of each local search using the Improvement Index (*I.I*) measure, and (2) the final results obtained by each MA are analyzed and are also compared against those obtained by the εDEga [14], the most competitive approach in the aforementioned special session. The parameter values used for each algorithm are described in Table 5.1. The dimensionalities used in the test problems were $D = 10$ and $D = 30$, as defined in the special session. Likewise, the maximum number of fitness evaluations *maxFEs* (200,000 and 600,000 respectively). The parameters marked with (*)

Algorithm 18 Memetic DE

Randomly generate an initial population of vectors $P_0 = (X_{0,i}, \ldots, X_{0,Pmax})$
Calculate the fitness of each vector in the initial population.
Set the ε value using Equation 3.13
repeat
  for $i \leftarrow 1, Pmax$ **do**
    Randomly select $r0$,$r1$,$r2 \in [1, Pmax]$ and $r0 \neq r1 \neq r2 \neq i$
    Randomly select $J_{rand} \in [1, D]$
    for $j \leftarrow 1, D$ **do**
      if $rand_j \leq Cr$ Or $j = Jrand$ **then**
        $u_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j})$
      else
        $u_{G,i,j} = x_{G,i,j}$
      end if
    end for
    if $U_{G,i} \leq_{\varepsilon} X_{G,i}$ using Equation 3.11 **then**
      $X_{G+1,i} = U_{G,i}$
    else
      $X_{G+1,i} = X_{G,i}$
    end if
  end for
  if No improvement counter $\geq T$ **then**
    Reset the no improvement counter
    Calculate ψ value using Equation 5.1
    if rand(0,1) $\leq \psi$ **then**
      Set $X_{new} \leftarrow$ **Local_Search_Operator**($X_{G,best}$)
      Set $X_{G,rand} \leftarrow X_{new}$ where $X_{G,rand} \neq X_{G,best}$
    end if
  end if
  if There are equality constraints **then**
    Modify δ using Equation 5.2
  end if
  Update ε value using Equation 3.12
  $G = G + 1$
until *MaxFEs* is reached

TABLE 5.1: Parameters values. (*) Indicates the values obtained using IRACE.

Algorithm	**Parameter**	**Value**	
		10 D	30 D
DE	*maxFEs*	$2.0E+05$	$6.0E+05$
	Pmax	20	50
	**Cr*	0.7890	0.8830
	**F*	0.8587	0.9989
	T	D	
Nelder-Mead	$*\beta$	0.7683	
	$*\gamma$	1.2030	
	maxIter	D×20	
Hooke-Jeeves	$*\Delta$	0.6986	
	$*\alpha$	1.0001	
	maxIter	D×2	
Hill Climbing	*maxIter*	D×50	
Epsilon-Constraint	θ	0.2	
	cp	46	
	Gc	1600	
Equality Constraint Tolerance	δ	1.8E+100	
	dec	1.1002	

in Table 5.1 were fine-tuned by using the IRACE tool [133] with four representative test problems as a training set for the parameter tuning processes. The parameters suggested in [14] and [5] were used as the starting point for such process.

The results of the first experiment (analyzing the performance of local search using the Improvement Index (*I.I*) measure) are presented in Figures 5.1 and 5.2. In all cases the 95%-confidence Wilcoxon test showed significant differences with the exception of MDE-HJ and MDE-HC in 10D problems. The results suggest that MDE+HC has a better overall average performance in most problems with respect to MDE+NM (10D and 30D) and MDE+HJ (only in 30D). Only in test problem C13 for 10D, MDE+HC had a negative *I.I.* average performance. On the other hand, MDE+NM obtained positive I.I. values for 10D in only four test problems (C04, C05, C06 and C12) and was clearly outperformed by MDE+HC. It is important to remark MDE+NM's clear improvement from 10D to 30D (most negative I.I. values in 10D and most positive I.I.values in 30D). In contrast, MDE+HJ shows an opposite behavior, i.e. most positive I.I. values in 10D but most negative I.I. values in 30D.

The results of the second experiment are summarized in Table 5.2, where the best and average function values for each MA, besides the standard deviation values, are shown for 10D and 30D. The aforementioned results suggest that MDE+NM, MDE+HJ and MDE+HC are able to find feasible solutions for all test problems in 10D and 30D.

The results of the Wilcoxon test applied to the best and average overall results between εDEga vs MDE+NM, εDEga vs MDE+HJ and εDEga vs MDE+HC are presented in Table 5.3. It can be noted that no significant differences were obtained, i.e. the performance of the three MAs with local search operators based on direct search methods is similar to that observed by a MA with gradient based local search.

TABLE 5.2: Function values achieved by MDE+NM, MDE+HJ and MDE+HC for 10D-30D and comparison against εDEga. Boldface remarks those better results and Italic text remarks those similar results.

Function	MA	10D			30D		
		Best	**Avg**	**Std**	**Best**	**Avg**	**Std**
C01	εDEga	*-7.473104E-01*	**-7.470402E-01**	**1.323339E-03**	**-8.218255E-01**	**-8.208687E-01**	**7.103893E-04**
	MDE+NM	*-7.473104E-01*	-7.376742E-01	1.749557E-02	-8.218000E-01	-8.054422E-01	1.333897E-02
	MDE+HJ	-7.473103E-01	-7.421135E-01	6.315751E-03	-8.217591E-01	-7.998397E-01	2.413851E-02
	MDE+HC	-7.473102E-01	-7.340776E-01	1.360141E-02	-8.178500E-01	-7.815536E-01	2.848675E-02
C02	εDEga	-2.277702E+00	-2.258870E+00	2.389779E-02	-2.169248E+00	-2.151424E+00	1.197582E-02
	MDE+NM	*-3.166209E+00*	-7.253218E-01	2.475302E+00	-3.001695E+00	1.490355E-01	2.901242E+00
	MDE+HJ	*-3.166209E+00*	-3.140613E+00	6.333234E-02	-2.178478E+00	-2.071487E+00	8.594076E-02
	MDE+HC	*-3.166209E+00*	**-3.165129E+00**	**1.947029E-03**	**-3.157959E+00**	**-3.155341E+00**	**1.843985E-03**
C03	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	2.867347E+01	**2.883785E+01**	**8.047159E-01**
	MDE+NM	*0.000000E+00*	4.783895E-01	1.295485E+00	4.018871E+01	1.055579E+03	1.219142E+03
	MDE+HJ	*0.000000E+00*	1.265642E+00	3.983805E+00	2.338906E+01	8.851698E+02	1.734344E+03
	MDE+HC	*0.000000E+00*	5.545954E+01	1.862923E+02	**2.867076E-02**	7.440057E+01	1.639822E+02
C04	εDEga	-9.992345E-06	-9.918452E-06	1.546730E-07	4.698111E-03	8.162973E-03	3.067785E-03
	MDE+NM	-4.441070E+01	-4.441070E+01	**7.105427E-15**	*-4.086475E+01*	*-4.086475E+01*	*2.842171E-14*
	MDE+HJ	**-5.162324E+01**	**-4.501714E+01**	1.574523E+00	*-4.086475E+01*	*-4.086475E+01*	1.852869E-14
	MDE+HC	-4.441070E+01	-4.441070E+01	7.519677E-15	*-4.086475E+01*	*-4.086475E+01*	*2.842171E-14*
C05	εDEga	-4.836106E+02	-4.836106E+02	3.890350E-13	-4.531307E+02	-4.495460E+02	2.899105E+00
	MDE+NM	-6.074970E+02	-6.074970E+02	*2.273737E-13*	-6.038261E+02	*-6.038261E+02*	*3.410605E-13*
	MDE+HJ	**-6.187782E+02**	**-6.087846E+02**	3.260816E+00	**-6.046462E+02**	-6.038602E+02	1.638842E-01
	MDE+HC	-6.074970E+02	-6.074970E+02	*2.273737E-13*	-6.038261E+02	*-6.038261E+02*	*3.410605E-13*
C06	εDEga	-5.786581E+02	-5.786528E+02	3.627169E-03	-5.285750E+02	-5.279068E+02	4.748378E-01
	MDE+NM	-5.908091E+02	-5.908091E+02	*2.273737E-13*	*-5.908091E+02*	*-5.908091E+02*	2.501110E-13
	MDE+HJ	**-5.965922E+02**	**-5.914438E+02**	1.404629E+00	*-5.908091E+02*	*-5.908091E+02*	*2.273737E-13*
	MDE+HC	-5.908091E+02	-5.908091E+02	*2.273737E-13*	*-5.908091E+02*	*-5.908091E+02*	*2.273737E-13*
C07	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	**1.147112E-15**	**2.603632E-15**	**1.233430E-15**
	MDE+NM	*0.000000E+00*	6.378527E-01	1.461504E+00	1.104374E+01	7.450512E+01	4.632270E+01
	MDE+HJ	*0.000000E+00*	3.543983E+01	1.153287E+02	1.958496E+01	1.319716E+02	2.140775E+02
	MDE+HC	2.335028E-27	1.140278E+01	3.160441E+01	5.484654E-04	3.613771E+01	6.136745E+01
C08	εDEga	*0.000000E+00*	**6.727528E+00**	**5.560648E+00**	**2.518693E-14**	**7.831464E-14**	**4.855177E-14**
	MDE+NM	*0.000000E+00*	9.743229E+01	2.368199E+02	2.054448E+01	3.295516E+02	4.599177E+02
	MDE+HJ	*0.000000E+00*	1.872247E+02	4.925612E+02	2.580141E+01	2.406614E+03	3.473454E+03
	MDE+HC	2.886723E-26	2.083554E+02	4.002769E+02	1.031884E-03	7.015442E+02	2.623504E+03
C09	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	**2.770665E-16**	**1.072140E+01**	**2.821923E+01**
	MDE+NM	*0.000000E+00*	3.189263E-01	1.081532E+00	3.076869E+01	1.300074E+05	5.463461E+05
	MDE+HJ	*0.000000E+00*	5.538516E+03	2.675761E+04	2.645223E+01	7.286739E+02	9.047144E+02
	MDE+HC	*0.000000E+00*	2.894271E+01	8.247018E+01	2.391410E-02	2.178574E+02	3.038142E+02
C10	εDEga	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**	3.252002E+01	**3.326175E+01**	**4.545577E-01**
	MDE+NM	*0.000000E+00*	7.973158E-01	1.594632E+00	3.297857E+01	1.319775E+04	4.578728E+04
	MDE+HJ	*0.000000E+00*	5.444295E+02	2.593653E+03	2.352233E+01	1.151545E+03	3.730467E+03
	MDE+HC	1.910937E-26	3.539641E+01	1.016156E+02	**1.229880E-02**	1.024342E+02	1.922225E+02
C11	εDEga	-1.522713E-03	-1.522713E-03	**6.341035E-11**	-3.268462E-04	-2.863882E-04	**2.707605E-05**
	MDE+NM	-7.164915E+01	-5.657112E+01	1.730193E+01	-6.043522E+01	**-5.458203E+01**	3.540663E+00
	MDE+HJ	-6.579135E+01	-5.585565E+01	5.458492E+00	**-6.044491E+01**	-5.048541E+01	4.155694E+00
	MDE+HC	**-7.205104E+01**	**-6.013852E+01**	5.316331E+00	-5.535454E+01	-5.016513E+01	3.314065E+00
C12	εDEga	-5.700899E+02	-3.367349E+02	**1.782166E+02**	-1.991453E-01	3.562330E+02	**2.889253E+02**
	MDE+NM	-8.108057E+03	**-7.932093E+03**	2.062272E+02	**-2.680487E+04**	**-2.575254E+04**	4.882107E+02
	MDE+HJ	**-8.158429E+03**	-7.781861E+03	3.168890E+02	-2.595765E+04	-2.423713E+04	8.386515E+02
	MDE+HC	-8.108057E+03	-7.683042E+03	3.521029E+02	-2.620216E+04	-2.425797E+04	1.106711E+03
C13	εDEga	*-6.842937E+01*	**-6.842936E+01**	**1.025960E-06**	**-6.642473E+01**	**-6.535310E+01**	**5.733005E-01**
	MDE+NM	*-6.842937E+01*	-6.365608E+01	2.530104E+00	-6.582004E+01	-6.406239E+01	1.114729E+00
	MDE+HJ	*-6.842937E+01*	-6.222556E+01	3.593505E+00	-6.431489E+01	-5.933518E+01	2.580831E+00
	MDE+HC	-6.842936E+01	-6.391025E+01	3.071028E+00	-6.432779E+01	-6.059491E+01	2.539741E+00
C14	εDEga	*0.000000E+00*	0.000000E+00	**0.000000E+00**	**5.015863E-14**	**3.089407E-13**	**5.608409E-13**
	MDE+NM	*0.000000E+00*	**4.411768E+06**	1.092644E+07	4.444056E+04	1.661360E+09	4.741723E+09
	MDE+HJ	*0.000000E+00*	6.598579E+06	1.407676E+07	2.305974E+01	1.018126E+04	2.453777E+04
	MDE+HC	7.187323E-11	1.375070E+11	5.419730E+11	2.797877E-03	5.547401E+07	2.677654E+08
C15	εDEga	**0.000000E+00**	1.798978E-01	**8.813156E-01**	**2.160345E+01**	**2.160376E+01**	**1.104834E-04**
	MDE+NM	1.922937E+12	**8.456202E+13**	7.011401E+13	5.919372E+13	1.932704E+14	6.761978E+13
	MDE+HJ	8.967169E+12	8.884936E+13	7.827173E+13	1.200730E+14	2.396448E+14	6.652184E+13
	MDE+HC	4.017477E+12	8.161327E+13	5.158264E+13	1.269401E+14	1.989160E+14	4.724277E+13
C16	εDEga	*0.000000E+00*	3.702054E-01	3.710479E-01	**0.000000E+00**	**2.168404E-21**	**1.062297E-20**
	MDE+NM	*0.000000E+00*	**1.808242E-01**	3.367572E-01	1.549673E-05	5.951875E-02	2.012325E-01
	MDE+HJ	*0.000000E+00*	8.004045E-02	1.695592E-01	1.461790E-06	1.351913E-02	5.601275E-02
	MDE+HC	9.864672E-03	4.208551E-02	**2.854565E-02**	1.687228E-11	3.036562E-02	4.586211E-02
C17	εDEga	1.463180E-17	1.249561E-01	1.937197E-01	2.165719E-01	6.326487E+00	4.986691E+00
	MDE+NM	*0.000000E+00*	**4.719779E-29**	**1.640253E-28**	8.102202E-02	1.881183E+01	2.806569E+01
	MDE+HJ	*0.000000E+00*	9.972161E-02	3.219155E-01	2.330202E-02	3.317397E-01	3.520637E-01
	MDE+HC	1.669513E-27	1.847996E-08	6.937622E-08	**1.132108E-06**	**4.067197E-03**	**8.070527E-03**
C18	εDEga	3.731439E-20	**9.678765E-19**	**1.811234E-18**	1.226054E+00	8.754569E+01	1.664753E+02
	MDE+NM	1.104539E+02	4.661089E+03	5.214531E+03	1.047617E-02	3.358028E+02	6.557928E+02
	MDE+HJ	2.088774E-07	1.083765E+02	1.802146E+02	1.769504E-02	**6.777942E-01**	**1.014288E+00**
	MDE+HC	**8.293069E-22**	9.905441E-01	4.852655E+00	**7.578104E-04**	1.083834E+00	2.901766E+00

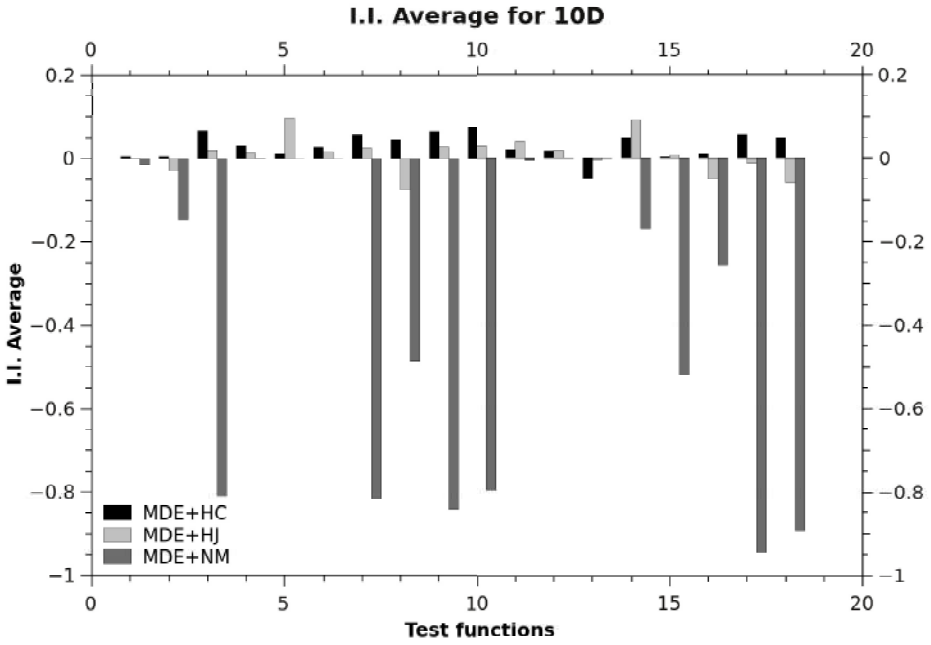


FIGURE 5.1: Average of improvement index (*I.I*) measure for 10D problems

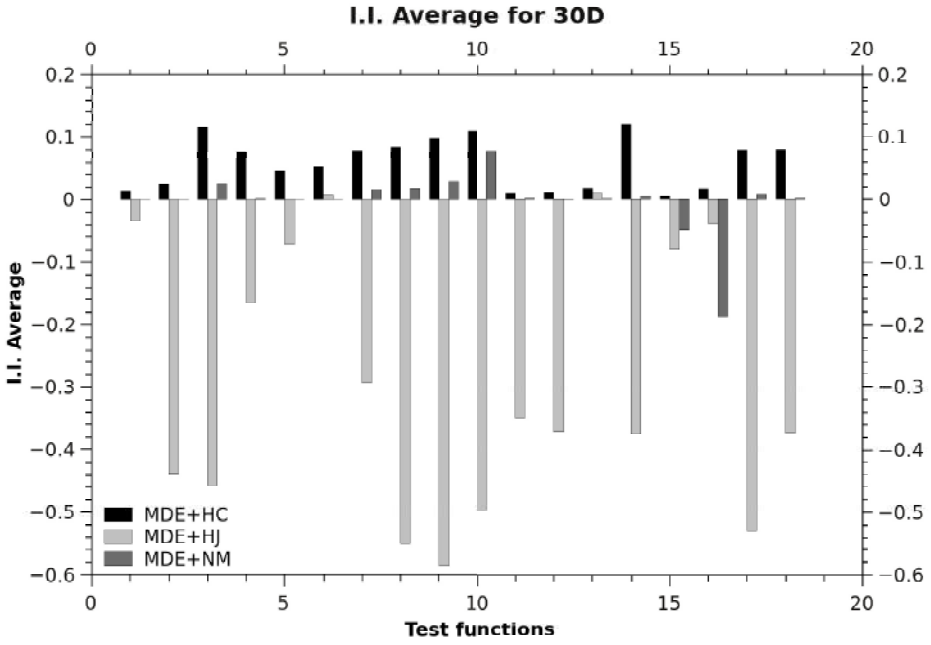


FIGURE 5.2: Average of improvement index (*I.I*) measure for 30D problems

Regarding comparisons in particular test problems, MDE+NM reaches the optimal function values obtained by εDEga for 10D in nine test problems (C01, C03, C07, C08, C09, C10, C13, C14 and C16). Furthermore, in functions C02, C04, C05, C06, C11, and C17, MDE+NM outperforms εDEga. For 30D, in seven test problems (C02, C04, C05, C11, C12, C17 and C18) MDE+NM gets better solutions than εDEga. Regarding MDE+HJ, it was able to outperform εDEga for 10D in seven test problems (C02, C04, C05, C06, C11, C12 and C17), and in C03, C07, C08, C09 C10, C13, C14

TABLE 5.3: 95%-confidence Wilcoxon rank-sum statistical test results between εDEga and each MA compared for 10D and 30D test problems. (y) means significant difference and (n) means no significant difference.

Comparison		Wilcoxon Test Results	
$p \leq 0.05$.		10 D	30 D
	Best	N	N
εDEga vs MDE+NM	Average	N	N
	Best	N	N
εDEga vs MDE+HJ	Average	N	N
	Best	N	N
εDEga vs MDE+HC	Average	N	N

and C16 the optimal functions values are reached. For 30D, MDE+HJ outperform εDEga in ten test problems (C02, C03, C04, C05, C06, C10, C11, C12, C17 and C18). On the other hand, MDE+HC was able to outperform εDEga in eight test problems (C02, C04, C05, C06, C11, C12, C17 and C18) for 10D and 30D, adding C03 and C10 to the latter. However, unlike MDE+NM and MDE+HJ, MDE+HC reaches only two optimal function values for 10D (C03 and C09).

Finally, based on the 95%-confidence Wilcoxon test, no significant differences among the three MAs studied in this paper were obained.

Comparing the results of experiments 1 and 2, it can be noted that MDE+HC had the best performance based on *I.I.* for 10D and 30D test problems (Figures 5.1 and 5.2). However, the final results in Table 5.2 show that MDE+HC provided a similar performance with respect to MDE+NM and MDE+HJ.

On the other hand, MDE+NM had the worst performance in 10D problems based on the I.I. measure (see Figure 5.1), but obtained very competitive final results in the same 10D test problems (Table 5.2). MDE+NM improved its performance for 30D based on the I.I. measure values (see Figure 5.2), but such values were not better than those presented by MDE+HC. Nevertheless, the final results obtained by MDE+NM in 30D (Table 5.2) were similar than those obtained by MDE+HC.

Finally, MDE+HJ, which obtained similar results (based on the Wilcoxon test) with respect to MDE+HC based in the I.I. values for 10D in Figure 5.1, obtained similar final results in such dimension in Table 5.2. A different behavior was observed in 30D (Figure 5.2 and Table 5.2), where most negative I.I. values were obtained by MDE+HJ compared with the most positive I.I. values by MDE+HC, but similar final results were obtained by MDE+HJ and MDE+HC.

The results of both experiments imply that the positive effect of a local search operator in a constrained search space can not be only measured by the isolated improvement of a single solution. A possible way to deal with this issue is considering such improvement but with respect to the improvement of the whole population.

5.1.6 Conclusions of Study 1

The results obtained confirmed that the algorithm coordination proposed in this work is suitable to get competitive MAs to solve CNOPs with local search operators based on direct methods. The results also suggested that a poor value of the

improvement index measure does not necessarily reflect on also poor final results obtained by the MA in a constrained search space. Therefore, considering only the local search performance as a criterion for the coordination mechanism would not be the most convenient decision in memetic approaches based on DE to solve CNOPs.

5.2 Study 2: Local Search Depth Influence

Usually, the LSO's depth is defined by the number of iterations or using another termination criterion. Nevertheless, as it was pointed in the previous chapter, the definition of the LSO's depth is not usually analyzed, and motivates this study which is aimed to analize five direct search methods (Random Walk, Simulated Annealing, Nelder-Mead, Hooke-Jeeves and Hill Climbing) in order to measure its exploitation capabilities considering different search depths and different initial solution. Besides, the five LSOs are added separately, to a Memetic Differential Evolution (MDE) approach inspired in the previous study (Section 5.1), with the aim to measure the influence of the LSO search depth in the final results obtained by each MA variant.

5.2.1 Performance Measure

In order to measure the local search methods exploitation capabilities, the proximity rate measure is proposed, which is calculated by the Euclidean distances from the initial solution vector and the solution vector obtained by the LSO regarding the best-known solution vector, see Equation 5.3.

$$\kappa = 1 - \left| \frac{d_{X^*,\widehat{X}}}{d_{X^*,X}} \right| \tag{5.3}$$

where X^* is the best-known solution vector, while $\widehat{X}$ and X are the new vector obtained by the LSO and the initial vector, respectively. $d_{X,Y}$ is the Euclidean distance between two vectors, and is calculated by Equation 5.4.

$$d_{X,Y} = \sqrt{\sum_{i=1}^{D} (X_i - Y_i)^2} \tag{5.4}$$

where D is the number of decision variables, i.e., the vector dimension.

The Proximity Rate value κ could lie between 0 and 1. When $\kappa \approx 1$ indicates a high LSO exploitation capability. However, cases where $d_{X^*,X'} > d_{X^*,X}$ induce negative κ values, indicating a distancing from optimal vector and poor LSO exploitation capability.

5.2.2 Memetic DE Approach

The Memetic Differential Evolution approach (MDE) implemented in this work consists of three main issues. (1) Exploitation area: The best solution in the population was used to exploit promising areas in the search space. (2) Application frequency:

inspired in [87] the LSO is activated by using a probability Φ (user-defined parameter) during the DE generations after the selection process. (3) Type of replacement: according to [134] Lamarckian learning is used, i.e., the solution generated by the LSO (X_{new}) is always kept for the next generation.

5.2.3 Experiments and Results

The experiments are divided in three phases:

1. The local search methods described in Section 2.4 (HJ, HC, NM, RW and SA) are tested on twenty four benchmark problems, described in Table A.1, used in the special session on "Single Objective Constrained Real-Parameter Optimization" in CEC'2006 [135], to analize the local search exploitation capabilities, i.e., the "proximity rate" introduced in Section 5.2.1). The aforementioned benchmark was selected, because the optimal solution vectors are reported in [135] and they are required to compute the "proximity rate" measure.
2. The final results by each LSO are analyzed by computing the 95% confidence Kruskal-Wallis test [135].
3. The final results of every MDE instance (MDE+HJ,MDE+HC, MDE+NM, MDE+RW and MDE+SA) are compared to asses the impact of the LSO exploitation capabilities.

In order to analyze the exploitation capabilities of direct local search methods in CNOPs, a set of initial vectors were randomly generated and divided in two groups: (1) feasibles and (2) infeasibles, each subset containing 50 vectors. Due to the fact that some LSO perform stochastic processes; 30 independent runs were computed per each initial vector in the twenty four test problems. The parameter values used for each algorithm are described in Table 5.4. Furthermore, four maximum number of fitness evaluations *maxFEs* were allowed, in order to analyze each LSO varying the depth (200, 500, 900, 1500 and 2000). The parameters in Table 5.4 were fine-tuned by using the IRACE tool [136] with six representative test problems as a training set for the parameter tuning processes, the parameters suggested in [5] and *maxFEs* = 1000 for each LSO were used as the starting point for such process. Hill Climbing (HC) implemented in this work does not require user-defined parameters, therefore such LSO was not considered for parameter tuning.

The results of the first experiment are depicted in Figure 5.3 where the number of fitness evaluations allowed for each LSO are presented for the subset of infeasible and feasible initial solutions. Despite increasing the maximum number of fitness evaluations for HC and HJ, they do not improve the proximity rate on feasible and infeasible initial solutions. On the other hand, HC has better exploitation capabilities in both subsets. The statistics values of the first experiment depicted in Figure 5.4 show greater consistency in results of HJ and SA than the other LSOs

The results of the second experiment are depicted in Figures 5.5 and 5.6, where the Kruskal-Wallis test suggests a significant difference between HJ and SA for infeasible initial solutions and between HC and SA for feasible initial solutions. The lines to the right hand-side mean a better exploitative capacity of the local search operator. Therefore, according to this analysis, HC and HJ are LSOs that have better exploitation capability of the search space, regardless of whether the initial search point is in a feasible region or not.

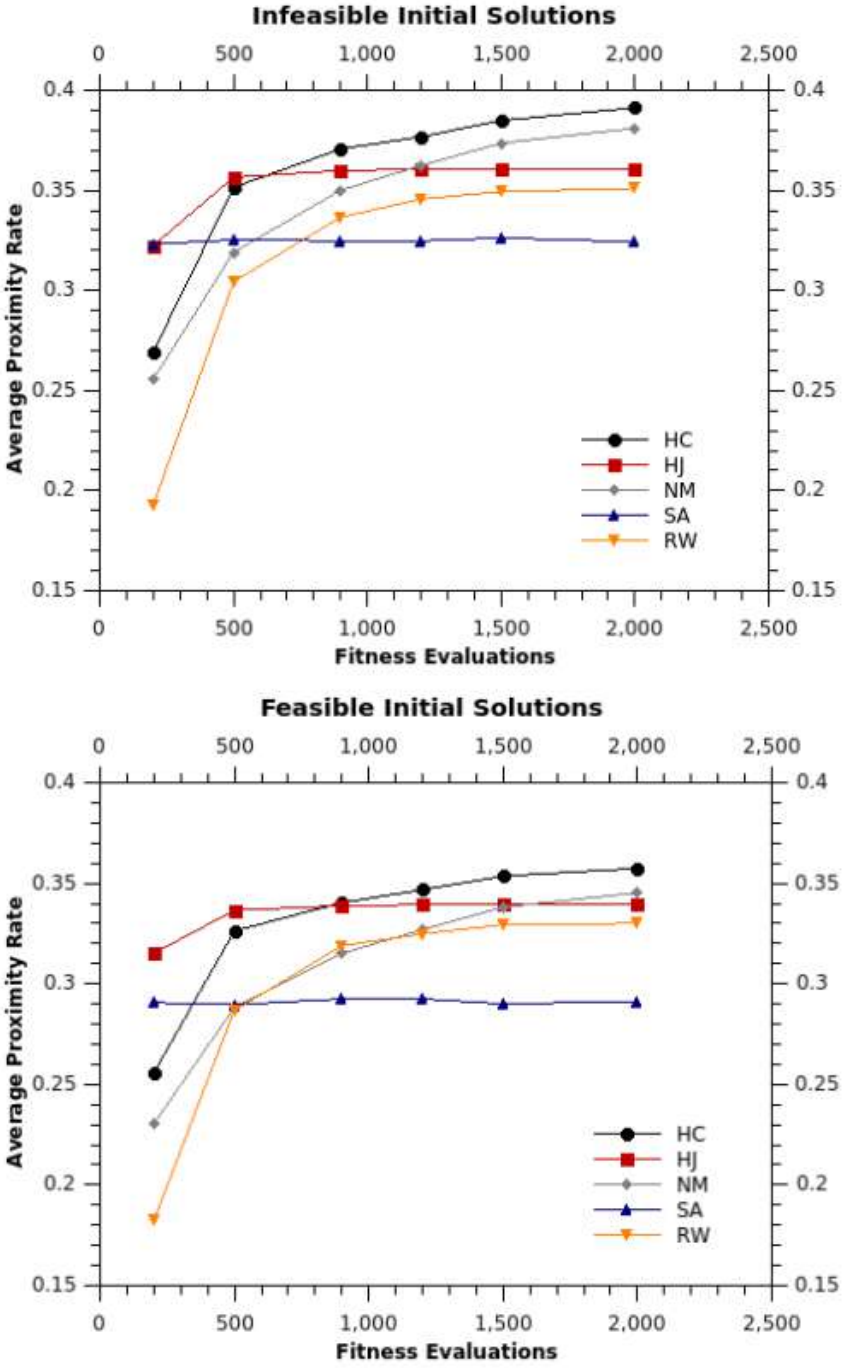


FIGURE 5.3: Proximity rate average for the 24 problems starting with 50 infeasible and 50 feasible solution vectors using different maximum number of fitness evaluations values (200, 500, 900, 1200, 1500 and 2000) as stop criterion for each local search (RW, SA, NM, HJ, HC).

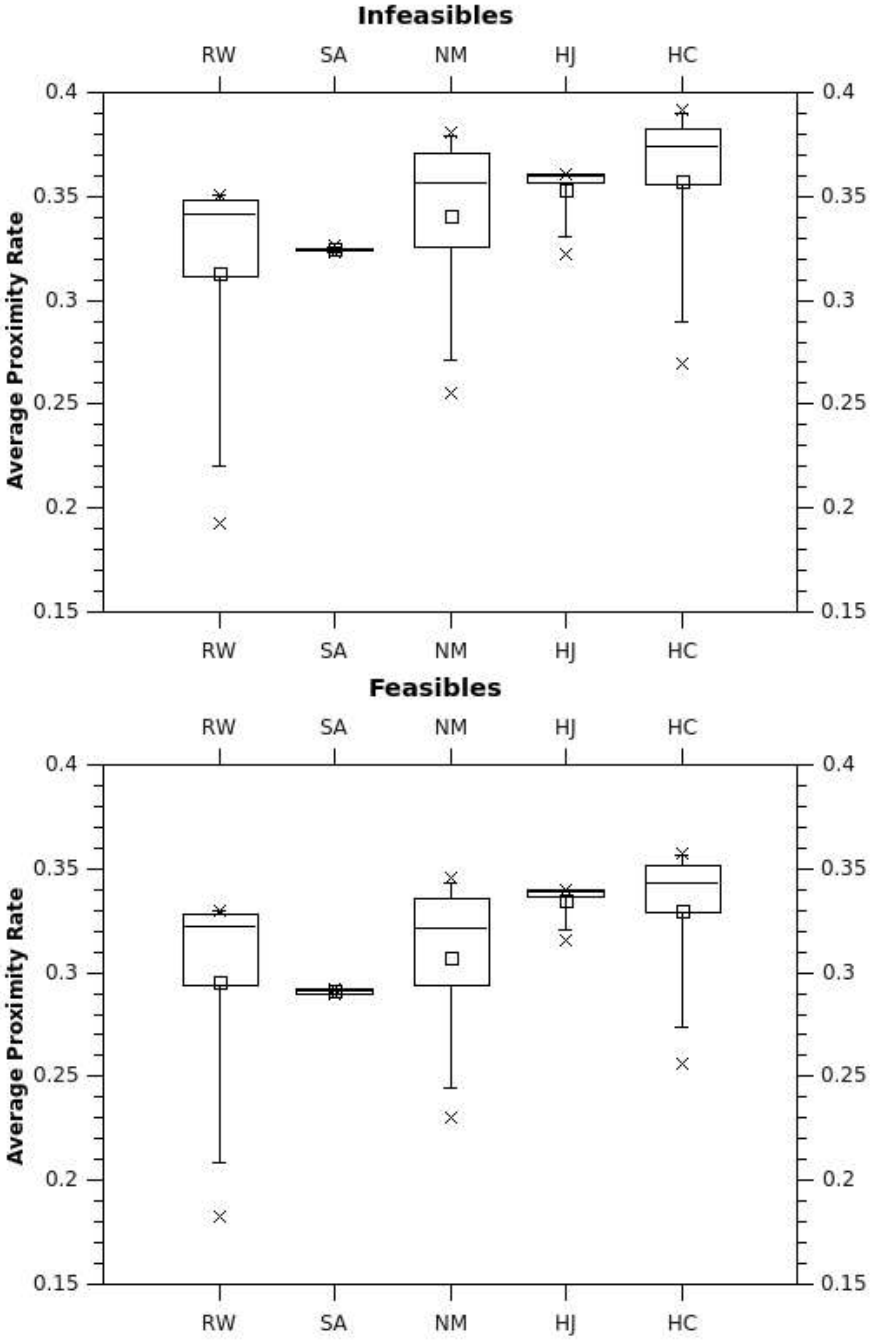


FIGURE 5.4: Box plot of proximity rate average for the 24 problems starting with 50 infeasible and 50 feasible solution vectors using different maximum number of fitness evaluations values (200, 500, 900, 1200, 1500 and 2000) as stop criterion for each local search (RW, SA, NM, HJ, HC).

TABLE 5.4: User-defined parameters values for Direct Local Search Methods. Φ is the probability of local search activation

Local Search	**Parameter**	**Value**
Random Walk	P	68
	z_0	5.96
	ε	0.18
Simulated Annealing	T	6603
	ε	0.56
Nerder-Mead	β	0.81
	γ	3.95
Hooke-Jeeves	Δ	1.79
	α	3.58
MDE	*maxFEs*	5.0E+5
	Pmax	100
	Cr	1.0
	F	0.55
	Φ	1.618

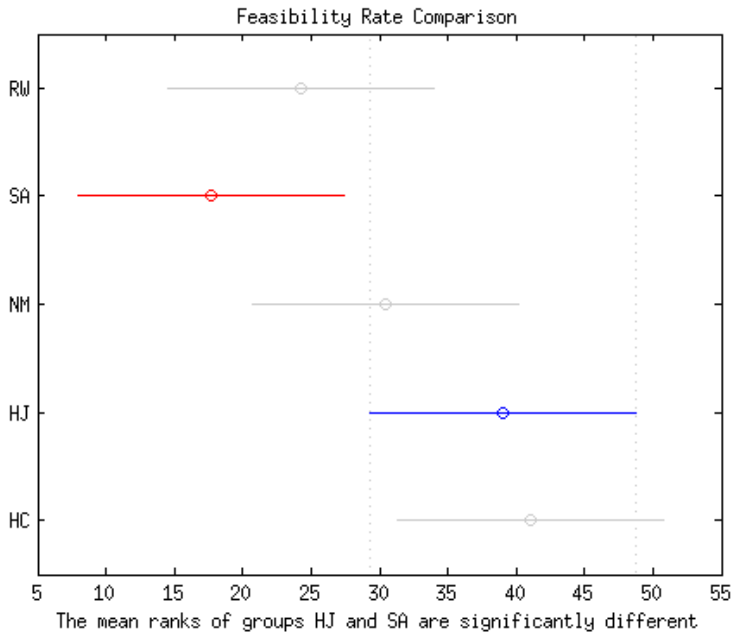


FIGURE 5.5: Kruskal-Wallis test result. Y-axis shows the LSO for **infeasible initial solutions**. The blue line shows significant difference against red line(s). Gray lines show no significant difference against the blue line

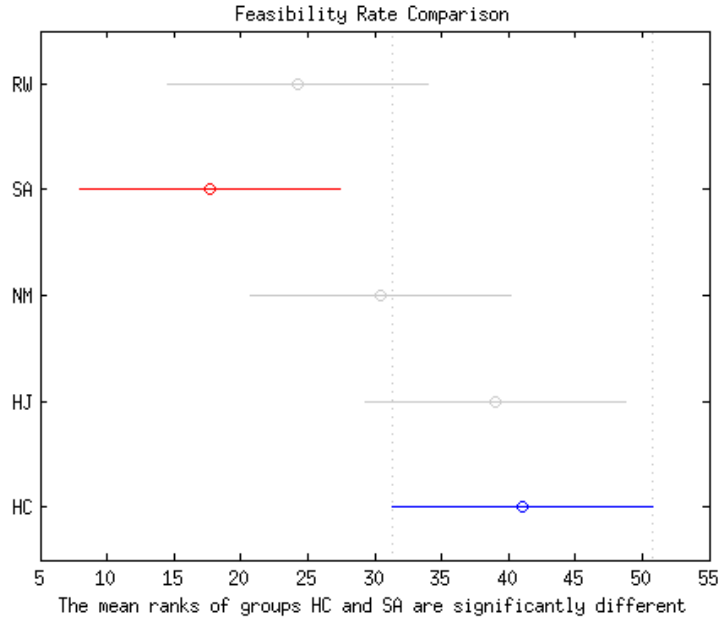


FIGURE 5.6: Kruskal-Wallis test result. Y-axis shows the LSO for **feasible initial solutions**.The blue line shows significant difference against red line(s). Gray lines show no significant difference against the blue line

The results of the third experiment are depicted in Table 5.5 where the best, mean and standard deviation function values achieved by each MDE (MDE+HJ, MDE+HC, MDE+NM, MDE+RW, MDE+SA) are presented and compared against the best-known fitness values reported in [135]. Despite the fact that the results presented in Table 5.5 were obtained using a low search depth by every LSO (500 maximum of fitness evaluations), all MDE approaches were able to achieve optimum values in most test problems, except in g13 test function.

5.2.4 Conclusions of Study 2

Numerical results showed that in spite of differences between the LSO behaviors in the test groups by using different search depths, the MDE final results were not affected by using shallow search depth; since in the most test problems, optimal values were achived and competitve results were obtained. Therefore, spending too many fitness evaluations during the search process of a direct LSO, is not required within MDE approaches that use a probabilistic global-local search coordination scheme to solve the CNOPs included in the benchmark adopted.

5.3 Study 3: Baldwing Effect vs Lamarckian Learning

In addition to exploitation area and the frequency of LSO activation, there are two basic models of evolution that can be used to incorporate learning into a DE: the Baldwin effect and Lamarckian learning, see Section 4.1. In the specialized literature, most of MDEs to solve CNOPs implements the Lamarckian learning [14, 16, 17, 19, 20, 79, 81, 137, 138, 139]. Whereas a synergy between Lamarckian and Baldwinian

TABLE 5.5: Function values achieved by MDE+HJ, MDE+HC, MDE+NM, MDE+RW, MDE+SA. Boldface remarks those similar results.

Function	Optimal	Criteria	MDE+HJ	MDE+HC	MDE+NM	MDE+RW	MDE+SA
g01	-15	Best	**-15**	**-15**	**-15**	**-15**	**-15**
		Mean	-14.999999999	-14.9999999996	-14.9999999994	-14.9999999995	-14.9999999995
		Std	2.73067054022752E-009	2.71496942377722E-010	8.14617567927385E-010	4.82788754785216E-010	5.45313835859241E-010
g02	-0.8036191042	Best	**-0.8036191042**	**-0.8036191042**	**-0.8036191042**	**-0.8036191042**	**-0.8036191042**
		Mean	-0.8036191042	-0.8036191042	-0.8036191042	-0.8036191042	-0.8036191042
		Std	0	0	0	0	0
g03	-1.0005001	Best	-0.6074561635	-0.6783584009	-0.7470916691	-0.6597960377	-0.7041044302
		Mean	-0.3877588788	-0.3967701438	-0.4322221701	-0.3704372768	-0.3671288173
		Std	0.0845512893	0.1017887439	0.1217925371	0.0752663858	0.1000313058
g04	-30665.5386717834	Best	**-30665.5386717833**	**-30665.5386717833**	**-30665.5386717833**	**-30665.5386717833**	**-30665.5386717833**
		Mean	-30665.5386717833	-30665.5386717833	-30665.5386717833	-30665.5386717833	-30665.5386717833
		Std	0	0	0	0	0
g05	5126.4967140071	Best	**5126.4967140071**	**5126.4967140071**	**5126.4967140071**	**5126.4967140071**	**5126.4967140071**
		Mean	5129.5819507654	5128.7644425589	5132.9200114817	5134.5405511058	5135.5597284402
		Std	6.9074646254	8.9375793271	15.7495576451	15.6995619604	22.0100714081
g06	-6961.8138755802	Best	**-6961.8138755802**	**-6961.8138755802**	**-6961.8138755802**	**-6961.8138755802**	**-6961.8138755802**
		Mean	-6961.8138755802	-6961.8138755802	-6961.8138755802	-6961.8138755802	-6961.8138755802
		Std	0	0	0	0	0
g07	24.3062090681	Best	**24.3062090682**	**24.3062090682**	**24.3062090682**	**24.3062090682**	**24.3062090682**
		Mean	24.3062090682	24.3062090682	24.3062090682	24.3062090682	24.3062090682
		Std	0	0	0	0	0
g08	-0.0958250415	Best	**-0.0958250414**	**-0.0958250414**	**-0.0958250414**	**-0.0958250414**	**-0.0958250414**
		Mean	-0.0958250414	-0.0958250414	-0.0958250414	-0.0958250414	-0.0958250414
		Std	0	0	0	0	0
g09	680.6300573745	Best	**680.6300573744**	**680.6300573744**	**680.6300573744**	**680.6300573744**	**680.6300573744**
		Mean	680.6300573744	680.6300573744	680.6300573744	680.6300573744	680.6300573744
		Std	0	0	0	0	0
g10	7049.2480205286	Best	**7049.2480205287**	**7049.2480205287**	**7049.2480205287**	**7049.2480205287**	**7049.2480205287**
		Mean	7049.2480205287	7049.2480205287	7049.2480205287	7049.2480205287	7049.2480205287
		Std	0	0	0	0	0
g11	0.7499	Best	**0.7499**	**0.7499**	**0.7499**	**0.7499**	**0.7499**
		Mean	0.7709333858	0.7653993034	0.763313252	0.7601100589	0.7658845001
		Std	0.0309807057	0.0313538681	0.0233864928	0.022847417	0.0379289094
g12	-1	Best	**-1**	**-1**	**-1**	**-1**	**-1**
		Mean	-1	-1	-1	-1	-1
		Std	0	0	0	0	0
g13	0.053941514	Best	0.1174305777	0.1338389892	0.0803370537	*0.062475734*	*0.0667524527*
		Mean	0.3759206198	0.4681336834	0.4008787459	0.4145928883	0.365800367
		Std	0.1698799607	0.1262920802	0.1165370953	0.1955497953	0.1477637386
g14	-47.7648884595	Best	**-47.7648287542**	**-47.7648287542**	**-47.7648287542**	**-47.7648287542**	**-47.7648287542**
		Mean	-47.7648287542	-47.7648287542	-47.7648287542	-47.7648287542	-47.7648287542
		Std	0	0	0	0	0
g15	961.7150222899	Best	**961.71502229**	**961.71502229**	**961.71502229**	**961.71502229**	**961.71502229**
		Mean	961.9639609687	961.8962403026	961.9081554909	961.8209591344	961.993228275
		Std	0.6986238731	0.5010224502	0.8744877736	0.2415972792	0.6442962331
g16	-1.9051552586	Best	**-1.9051552585**	**-1.9051552585**	**-1.9051552585**	**-1.9051552585**	**-1.9051552585**
		Mean	-1.9051552585	-1.9051552585	-1.9051552585	-1.9051552585	-1.9051552585
		Std	0	0	0	0	0
g17	8853.5396748064	Best	**8853.5396748065**	**8853.5396748065**	**8853.5396748065**	**8853.5396748065**	**8853.5396748065**
		Mean	8925.4961384738	8931.0117172956	8927.6478801476	8925.9449094633	8938.341490254
		Std	20.7031235649	3.637864142	13.5926402527	19.7526905967	41.9359621537
g18	-0.8660254038	Best	**-0.8660254038**	**-0.8660254038**	**-0.8660254038**	**-0.8660254038**	**-0.8660254038**
		Mean	-0.8660254038	-0.8660254038	-0.8660254038	-0.8660254038	-0.8660254038
		Std	0	0	0	0	0
g19	32.6555929502	Best	**32.6555929502**	**32.6555929502**	**32.6555929502**	**32.6555929502**	**32.6555929502**
		Mean	32.6555929502	32.6555929502	32.6555929502	32.6555929502	32.6555929502
		Std	1.52554012876978E-012	1.73479076935423E-012	2.16221309422939E-012	2.90800979031505E-012	4.16578225753484E-012
g21	193.72451007	Best	**193.7245100697**	**193.7245100697**	**193.7245100697**	**193.7245100697**	**193.7245100697**
		Mean	227.6818553498	266.4902499555	242.2350033269	242.2350033269	256.7881513041
		Std	58.4917905939	66.3234564215	64.4548139823	64.4548139823	66.6909022052
g23	-400.0551	Best	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**	**-400.0551**
		Mean	-400.0551	-400.0551	-400.0551	-400.0551	-400.0551
		Std	0	0	0	0	0
g24	-5.5080132716	Best	**-5.5080132716**	**-5.5080132716**	**-5.5080132716**	**-5.5080132716**	**-5.5080132716**
		Mean	-5.5080132716	-5.5080132716	-5.5080132716	-5.5080132716	-5.5080132716
		Std	0	0	0	0	0

learning was proposed in [140] for a distributed MDE for unconstrained numerical optimization problem.

The Baldwin effect in MDE for CNOPs is not usually analyzed. This is the main motivation for this study which is aimed to analyze the Baldwin effect into an MDE that incorporates the Hooke-Jeeves method as local search operator (MDEHJ). MDEHJ adopts the ε-constrained method [62] as constraint-handling mechanism.

5.3.1 Memetic DE Approach

The integration of HJ within the standard DE is based on the fact that DE/rand/1/bin has a good performance regarding exploration capabilities in constrained spaces [141], and considering that for some problems DE keeps trapped in local optima [10]. In this study, HJ is performed after the selection operator of DE under a dynamic probabilistic scheme described in this section. Furthermore, three issues were considered in the global-local search coordination: (1) exploitation area, (2) local search activation mechanism and (3) type of learning. The complete memetic DE named $MDEHJ$ is defined in Algorithm 19, where the gray lines mark the elements to integrate HJ as local search operator.

Exploitation Area In order to analyze the Baldwin effect within the Memetic DE, three different exploitation areas during the search process are considered in this study, and implemented in three instances of MDEHJ, separately.

1. Best area $MDEHJ_{best}$. HJ takes the best vector of the population as initial search point.
2. Worst area $MDEHJ_{worst}$. HJ takes the worst vector of the population as initial search point.
3. Random area $MDEHJ_{rand}$. HJ takes a random vector of the population as initial search point.

For this process, Line 24 in Algorithm 19 (Set j *) is replaced by: (a) Set $j \leftarrow$ best, (b) Set $j \leftarrow$ worst, and (c) Set $j \leftarrow$ rand for $MDEHJ_{best}$, $MDEHJ_{worst}$, and $MDEHJ_{rand}$, respectively.

Local Search Activation Mechanism Considering the fact that an excessive use of the local search in a memetic algorithm is not beneficial for the search process [25]; in this proposal, the local search is handled by means of a probability of activation, which includes the population diversity information in terms of the objective function values. Inspired in [142] a sinusoidal function is used to modulate the probability of HJ activation. See Equation 5.5.

$$P = p_{min} * (sin(2\pi * freq * G) * \chi + 1) \tag{5.5}$$

where p_{min} is the minimum probability allowed to be applied the local search, $freq$ is the frequency of probability variation, while G is the current generation in the global search cycle. Finally, χ (see Equation 5.6) is an estimation of the best vector performance with respect to the other vectors [28].

$$\chi = \frac{|f_{best} - f_{avg}|}{max|f_{best} - f_{avg}|_G} \tag{5.6}$$

where f_{best} and f_{avg} are the objective function values of the best and average solutions of the population, respectively. $max|f_{best} - f_{avg}|_G$ is the maximum difference observed (e.g., at the G^{th} generation). The behavior expected by this mechanism is if the population tends to resemble the best perfomance vector, the probability of applying the local search P tends to p_{min}.

Type of Learning In this study, the Baldwinian Learning is considered, i.e., after local search activation, the perfomance of the initial vector (constraint violation sum $\phi(X_{G,j})$ and objective function value $f(X_{G,j})$) is replaced by the performance of the new vector. In order to compare the Baldwin effect, the Lamarckian Learning is implemented separately. For this process, Line 26 in Algorithm 19 (Apply Learning *) is replaced by Equation 5.7 and Equation 5.8 for Baldwinian and Lamarckian learning, respectively.

$$\text{Set } f(X_{G,j}) \leftarrow f(X_{new}), \quad \phi(X_{G,j}) \leftarrow \phi(X_{new}) \tag{5.7}$$

$$\text{Set } X_{G,j} \leftarrow X_{new}, \quad f(X_{G,j}) \leftarrow f(X_{new}), \quad \phi(X_{G,j}) \leftarrow \phi(X_{new}) \tag{5.8}$$

5.3.2 Experiments and Results

The experiments are divided in two phases: (1) Three instances of MDEHJ mentioned in Section 5.3.1 were tested on eighteen well-known benchmark problems [132], with different search space dimensionality, 10 and 30 dimensions, to analyze the behavior of the algorithm using the Baldwin effect and the Lamarckian learning. Statistical values on 25 independent runs were computed for each test problem (2) The final results obtained by each MDEHJ instance were compared against those obtained by εDEga [14], which is a representative algorithm of the state-of-the-art in the scope of Memetic DE for CNOPs. The parameter values used for each algorithm are described in Table 5.6. The dimensionalities used in the test problems were $D = 10$ and $D = 30$, as defined in [132]. Likewise, the maximum number of fitness evaluations *maxFEs* (200,000 and 600,000 respectively). The parameters were fine-tuned by using the IRACE tool [136] with nine representative test problems as a training set for the parameter tuning processes, the parameters suggested in [14] and [5] were used as the starting point for such process.

The results of the first phase of the experiment are summarized in Tables 5.8 and 5.9, where the best, mean, and median function values for each MDEHJ instance, besides the standard deviation values, are shown for 10D and 30D, respectively. In addition, the feasibility rate (FR) of the 25 independent runs is presented, see Equation 5.9.

$$FR = \frac{\text{number of feasible runs}}{\text{total runs}} \tag{5.9}$$

where a feasible run is a run during which at least one feasible solution is found.

The aforementioned results suggest that $MDEHJ_{best}$, $MDEHJ_{worst}$, and $MDEHJ_{rand}$ using the Baldwinian and Lamarckian learning are able to find feasible solutions for all test problems in 10D. However, for the three MDEHJ instances with Baldwinian learning, it was not able to converge to feasible solutions in the C11 problem in 30D, which has a flat landscape function. Meanwhile, by using the Lamarckian learning the three MDEHJ instances presented a suitable perfomance for all test problems in 30D.

Algorithm 19 Memetic Differential Evolution MDEHJ

Randomly generate an initial population of vectors $P_0 = (X_{0,i}, \ldots, X_{0,Pmax})$
Calculate the fitness of each vector in the initial population.
Set the ε value using Equation 3.13
repeat
  for $i \leftarrow 1, Pmax$ **do**
    Randomly select $r0$,$r1$,$r2 \in [1, Pmax]$ and $r0 \neq r1 \neq r2 \neq i$
    Randomly select $J_{rand} \in [1, D]$
    for $j \leftarrow 1, D$ **do**
      if $rand_j \leq Cr$ Or $j = Jrand$ **then**
        $u_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j})$
      else
        $u_{G,i,j} = x_{G,i,j}$
      end if
    end for
    if $U_{G,i} \leq_{\varepsilon} X_{G,i}$ using Equation 3.11 **then**
      $X_{G+1,i} = U_{G,i}$
    else
      $X_{G+1,i} = X_{G,i}$
    end if
  end for
  Set the population diversity χ using Equation 5.6

  Set the activation probability P Equation 5.5

  if $rand(0, 1) \leq P$
  then
    Set j *

    Set $X_{new} \leftarrow$ **Hooke-Jeeves**($X_{G,j}$), using Algorithm 2

    Apply Learning *
  end if
  Update ε value using Equation 3.12
  $G = G + 1$
until *MaxFEs* is reached

TABLE 5.6: Parameters values for Differential Evolution (DE), Hooke-Jeeves (HJ), constraint handler (ε-Constrained), and the local search activation mechanism (P)

Algorithm	**Parameter**	**Value**	
		10 D	30 D
DE	*maxFEs*	$2.0E+05$	$6.0E+05$
	Pmax	100	250
	Cr	0.9	
	F	0.55	
HJ	α	2.5	
	δ	0.8	
	maxFEs	900	
ε-Constrained	θ	0.75	
	cp	4.5	
	Gc	1100	
P	p_{min}	0.05	
	$freq$	χ, see Equation 5.6	

TABLE 5.7: 95%-confidence Wilcoxon rank sum test results for the MDEHJ instances with Baldwinian learning ($B-MDEHJ_{best}$, $B-MDEHJ_{worst}$ and $B-MDEHJ_{rand}$) against MDEHJ instances with Lamarckian learning ($L-MDEHJ_{best}$, $L-MDEHJ_{worst}$ and $L-MDEHJ_{rand}$). **Dim** means the dimension of results, while $w+$ and $w-$ mean the sum of positive and negative ranks, respectively. Finally, **Diff** denotes whether there is a significant difference.

Algorithms	**Dim**	**Criteria**	$w+$	$w-$	**Diff**
$B-MDEHJ_{best}$ to $L-MDEHJ_{best}$	10	Best Fitness	11	17	$\approx$
		Average Fitness	89	64	$\approx$
	30	Best Fitness	103	68	$\approx$
		Average Fitness	166	5	$-$
$B-MDEHJ_{worst}$ to $L-MDEHJ_{worst}$	10	Best Fitness	18	18	$\approx$
		Average Fitness	71	65	$\approx$
	30	Best Fitness	137	34	$-$
		Average Fitness	133	38	$-$
$B-MDEHJ_{rand}$ to $L-MDEHJ_{rand}$	10	Best Fitness	16	5	$\approx$
		Average Fitness	84	52	$\approx$
	30	Best Fitness	116	55	$\approx$
		Average Fitness	126	45	$\approx$

Regarding the feasibility rate for test instances at 10D $MDEHJ_{best}$, $MDEHJ_{worst}$, and $MDEHJ_{rand}$ using both learning methods, has similar perfomance. However, increasing the dimensionality of the problems to 30D, MDEHJ instances with Lamarckian learning outperforms the Baldwinian learning.

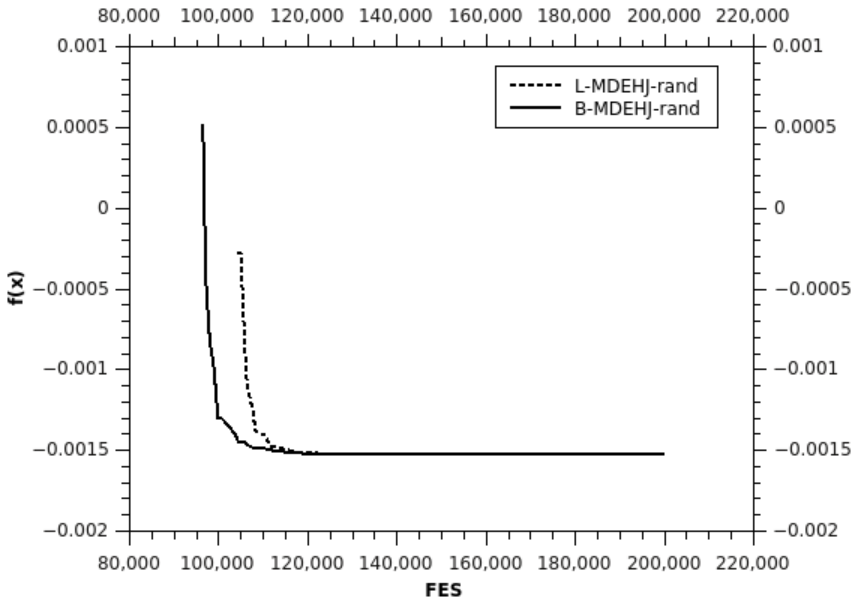
The convergence plots in Figure 5.7 for C11 and C17 test problems at 10D, show a similar behavior of the $MDEHJ_{rand}$ instances with Lamarckian learning ($L - MDEHJ_{rand}$) and Baldwinian learning ($B - MDEHJ_{rand}$). On the other hand, the convergence plots for the same test problems (C11 and C17) at 30D was presented in Figure 5.8, show a relative difference between both learning methods, where the performance of Lamarckian learning is better than the Baldwinian learning, considering that the plots show only feasible solutions of the median run out of the 25 independent runs.

Finally, in consideration of the similar final perfomance between the MDEHJ instances with Baldwinian and Lamarckian learning, the 95%-confidence Wilcoxon rank sum test was performed, in order to determine the statistical significant difference between approaches. The comparison is summarized in Table 5.7. One of the three signs ($+$, $-$, $\approx$) was assigned for the comparison, where "$+$" sign means the first algorithm is significantly better than the second, "$-$" sign means that the first algorithm significantly worse, and "$\approx$" sign means that there is no significant difference between the two algorithms.

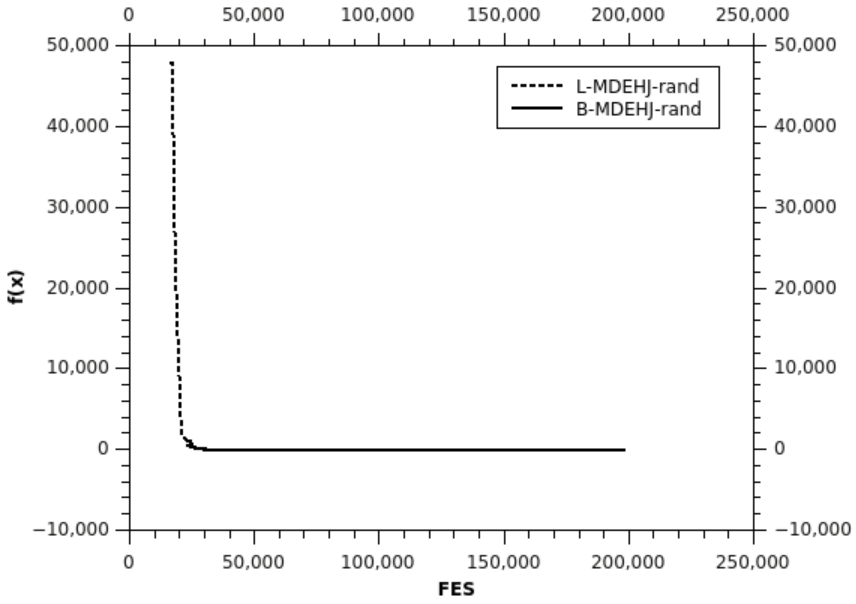
In the same way, the statistical comparison between $B-MDEHJ_{best}$, $B-MDEHJ_{worst}$, $B-MDEHJ_{rand}$, $L-MDEHJ_{best}$, $L-MDEHJ_{worst}$, $L-MDEHJ_{rand}$ and εDEga by the 95%-confidence Kruskall-Wallis test is shown in Figure 5.9.

5.3.3 Conclusions of Study 3

The results suggested that the algorithm proposed was suitable to solve constrained numerical problems and those results also showed that the use of Baldwinian Learning did not affect the final performance of the MDEHJ in constrained search spaces. Although there were numerical differences in some test functions, the statistical tests did not show significant differences with respect to the results reported by εDEga [14]. However, the Baldwin effect caused a deterioration in the quality of final results in larger dimensions. Finally, the results showed that the Baldwinian learning is not significantly affected by the area of exploitation of the local search operator.

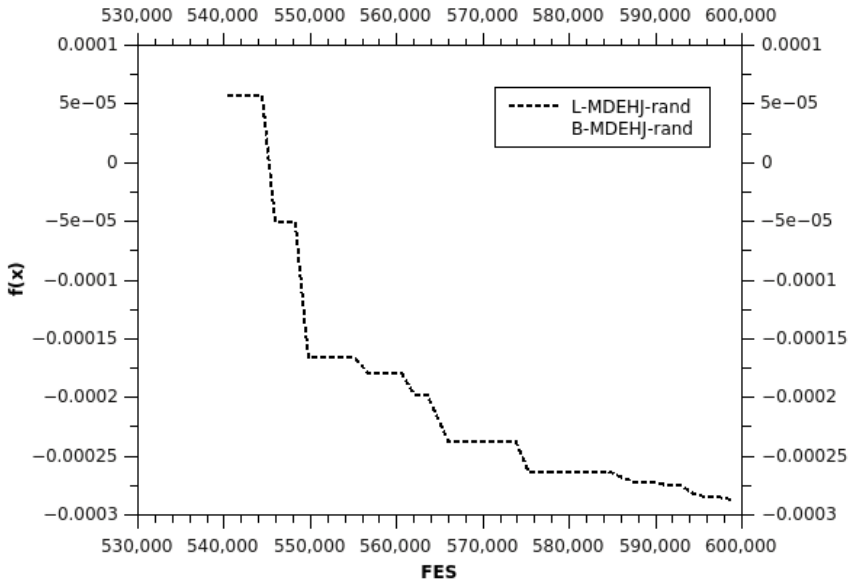


(A) C11

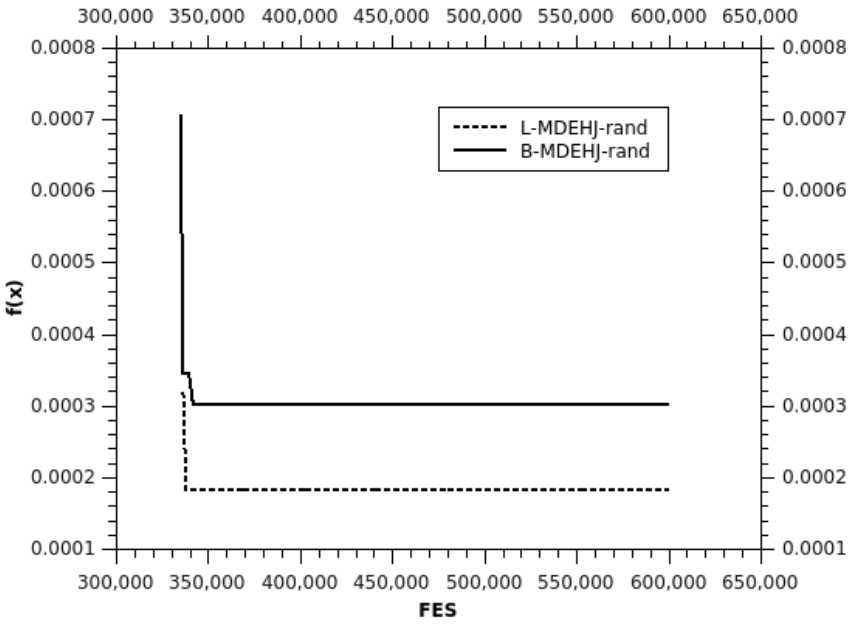


(B) C17

FIGURE 5.7: Convergence plot of the best run for C11 and C17 test problems in 10D

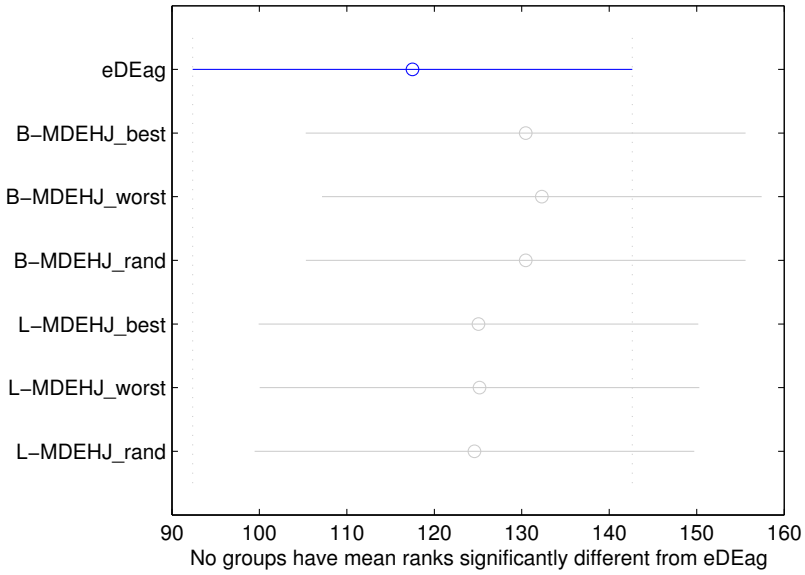


(A) C11

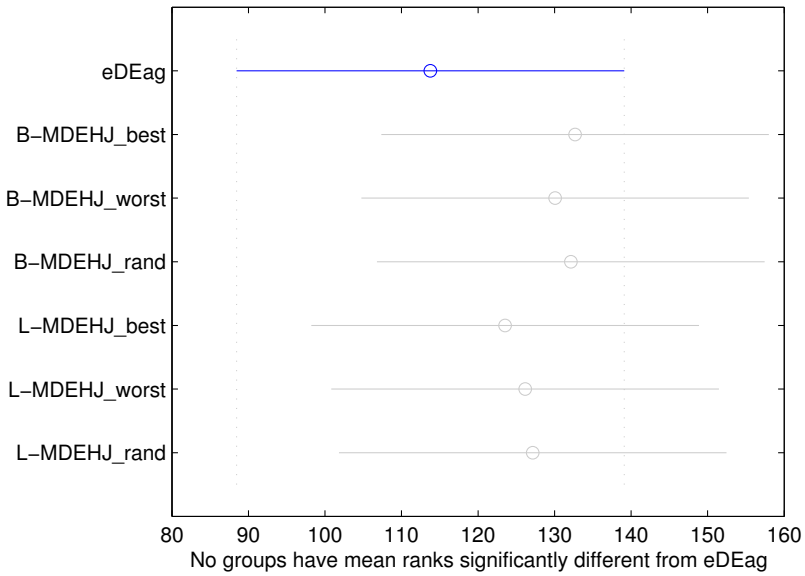


(B) C17

FIGURE 5.8: Convergence plot of the best run for C11 and C17 test problems in 30D



(A) Comparison of best fitness values



(B) Comparison of mean fitness value

FIGURE 5.9: 95%-confidence Kruskall-Wallis Test comparison of the best and mean fitness values of 25 runs for 36 test problems obtained by MDEHJ instances and values reported in the literature εDEag [14]

TABLE 5.8: Function values achieved by $MDEHJ_{best}$, $MDEHJ_{worst}$, and $MDEHJ_{rand}$ using the Baldwinian Learning (B-Learning) and the Lamarckian Learning (L-Learning) for 10D. Boldface remarks those better results and Italic text remarks those similar results. "Std" is the standard deviation and "FR" is the feasibility ratio

Problem	Criteria	B-Learning			L-Learning		
		$MDEHJ_{best}$	$MDEHJ_{worst}$	$MDEHJ_{rand}$	$MDEHJ_{best}$	$MDEHJ_{worst}$	$MDEHJ_{rand}$
C01	Best	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*
	Mean	**-7.457762E-01**	-7.415247E-01	-7.420458E-01	-7.431889E-01	-7.436112E-01	-7.419424E-01
	Median	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*	*-7.473104E-01*
	Std	0.0041098608	0.0109592183	0.0099075169	0.0107297238	0.0080123085	0.0103893801
	FR	1	1	1	1	1	1
C02	Best	*-2.277711E+00*	-2.258584E+00	-2.263124E+00	*-2.277711E+00*	-2.261749E+00	*-2.277711E+00*
	Mean	-1.842926E+00	-1.876020E+00	-1.947851E+00	-1.832271E+00	**-2.090904E+00**	-2.076175E+00
	Median	-1.917811E+00	-1.936744E+00	-1.947646E+00	-1.866249E+00	-2.085177E+00	**-2.122172E+00**
	Std	0.4396571011	0.2311764074	0.2090517645	0.3114133079	0.0949487515	0.1599606583
	FR	0.6	0.88	0.8	0.88	1	1
C03	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	8.646401E-21	1.633739E-15	6.808154E-08	8.296713E-27	**0.000000E+00**	6.105617E-11
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	4.14666837248954E-020	8.00365524192589E-015	3.40407692078578E-007	4.06454272443738E-026	0	2.11554194192275E-010
	FR	0.92	0.96	1	0.96	1	1
C04	Best	-9.907976E-06	-9.963639E-06	**-9.964802E-06**	-9.928846E-06	-9.846275E-06	-9.929543E-06
	Mean	-9.722134E-06	-9.187095E-06	-9.599487E-06	**-9.737280E-06**	-9.269496E-06	-9.478623E-06
	Median	-9.747407E-06	-9.655966E-06	-9.740492E-06	**-9.816262E-06**	-9.643077E-06	-9.550767E-06
	Std	1.56304926258468E-007	1.04189991296742E-006	4.47320987089503E-007	2.62613707406536E-007	7.48449324669952E-007	3.98643999703046E-007
	FR	1	1	1	1	1	1
C05	Best	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*
	Mean	-4.834632E+02	-4.834646E+02	-4.833612E+02	**-4.836106E+02**	-4.836046E+02	-4.835001E+02
	Median	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*	*-4.836106E+02*
	Std	0.6594188279	0.4375054037	1.0345232613	1.05666827861487E-006	0.0295881941	0.5414564094
	FR	0.8	0.96	0.92	1	0.96	0.96
C06	Best	-5.771759E+02	-5.771759E+02	-5.771759E+02	*-5.786624E+02*	*-5.786624E+02*	*-5.786624E+02*
	Mean	-5.771697E+02	-5.771757E+02	-5.771719E+02	-5.786111E+02	**-5.786489E+02**	-5.786253E+02
	Median	-5.771759E+02	-5.771759E+02	-5.771759E+02	-5.786618E+02	-5.786619E+02	**-5.786622E+02**
	Std	0.0234111508	0.0004340124	0.0149101872	0.1622863409	0.0471323892	0.0960564387
	FR	1	1	1	1	1	1
C07	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	3.189263E-01	*0.000000E+00*
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	0	0	0	0	1.1038337729	0
	FR	1	1	1	1	1	1
C08	Best	*0.000000E+00*	3.622903E+00	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	8.616990E+00	9.344484E+00	1.039322E+01	8.824209E+00	**6.954828E+00**	9.675175E+00
	Median	1.057285E+01	1.057285E+01	1.057285E+01	1.094154E+01	1.057285E+01	**1.057285E+01**
	Std	3.9306974057	2.8144402491	4.2434944831	4.1632269397	4.9438694851	3.5019303146
	FR	1	1	1	1	1	1
C09	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	2.278692E+02	**1.763261E-01**	3.526522E-01	5.285192E+00	6.070858E-01	6.070858E-01
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	1137.7409432408	0.8816305482	1.2205622579	19.0355223519	2.2930071102	2.2930071102
	FR	1	1	1	1	1	1
C10	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	**1.746854E-28**	8.984673E-01	3.274426E+02	1.924867E-01	2.064902E-01	6.979122E+00
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	8.734267942607 04E-028	3.6032326316	1637.2130644936	0.9624336178	1.0324509443	34.8956083253
	FR	1	1	1	1	1	1
C11	Best	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*
	Mean	**-1.522713E-03**	-1.519060E-03	-1.502779E-03	-1.522485E-03	-1.522641E-03	-1.522691E-03
	Median	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*	*-1.522713E-03*
	Std	7.31916988036191E-014	1.66394942968316E-005	8.046735928420 89E-005	1.09533781681381E-006	2.41098308573264E-007	8.67186715407771E-008
	FR	0.8	0.96	0.96	0.92	0.88	0.92
C12	Best	-4.265165E+02	**-5.683828E+02**	-3.054888E+02	-4.231332E+02	-4.231332E+02	-3.054888E+02
	Mean	**-2.704218E+02**	-1.905477E+02	-1.639763E+02	-2.404028E+02	-1.202660E+02	-1.528440E+02
	Median	-2.704218E+02	-1.726183E+02	-1.751087E+02	**-2.978758E+02**	-1.992458E-01	-1.528440E+02
	Std	220.7512439684	197.8587087849	125.3115670723	209.8387397247	185.9191539192	176.2589840633
	FR	0.08	0.28	0.72	0.24	0.2	0.16
C13	Best	-6.420751E+01	**-6.842937E+01**	-6.557847E+01	-6.557847E+01	-6.352529E+01	-6.557847E+01
	Mean	-6.183692E+01	-6.171669E+01	-6.239441E+01	-6.185219E+01	-6.233050E+01	**-6.254727E+01**
	Median	-6.227640E+01	-6.227640E+01	-6.227640E+01	-6.227640E+01	-6.227640E+01	**-6.351694E+01**
	Std	1.8902186454	3.0568101145	1.8993318778	2.1272165214	1.6532330626	2.4280705325
	FR	1	1	1	1	1	1
C14	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	4.605731E+00	7.682420E-01	**1.594632E-01**	3.189263E-01	5.635299E+00	5.201476E+00
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	13.3202584868	3.1142829337	0.7973158225	1.1038337729	21.4114747568	21.4113820039
	FR	1	1	1	1	1	1
C15	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	3.687537E+01	**1.798990E-01**	1.235283E+01	1.472207E+00	1.087769E+00	6.041938E-01
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	131.0444618988	0.899495008	47.226500902	6.4866240569	3.6840524499	2.2695237415
	FR	1	1	1	1	1	1
C16	Best	1.573042E-01	4.375902E-02	2.515554E-01	**0.000000E+00**	1.654749E-02	7.897881E-02
	Mean	7.182761E-01	**6.893921E-01**	7.419267E-01	7.603009E-01	7.466949E-01	7.822406E-01
	Median	**7.012325E-01**	7.025873E-01	8.104133E-01	1.006710E+00	9.558551E-01	9.592691E-01
	Std	0.3922117635	0.3221681271	0.2879725906	0.417739106	0.3712710438	0.3244318414
	FR	1	0.92	1	0.96	1	1
C17	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	4.930381E-34	*2.465190E-34*	*2.465190E-34*	4.930381E-34	*2.465190E-34*	4.930381E-34
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	1.70645073433125E-033	1.23259516440783E-033	1.23259516440783E-033	1.70645073433125E-033	1.23259516440783E-033	1.70645073433125E-033
	FR	1	1	1	1	1	1
C18	Best	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Mean	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Median	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	Std	0	0	0	0	0	0
	FR	1	1	1	1	1	1

TABLE 5.9: Function values achieved by $MDEHJ_{best}$, $MDEHJ_{worst}$, and $MDEHJ_{rand}$ using the Baldwinian Learning (B-Learning) and the Lamarckian Learning (L-Learning) for 30D. Boldface remarks those better results and Italic text remarks those similar results. "Std" is the standard deviation and "FR" is the feasibility ratio

Problem	Criteria	B-Learning			L-Learning		
		$MDEHJ_{best}$	$MDEHJ_{worst}$	$MDEHJ_{rand}$	$MDEHJ_{best}$	$MDEHJ_{worst}$	$MDEHJ_{rand}$
C01	Best	-8.212299E-01	-8.180021E-01	-8.189759E-01	**-8.218336E-01**	-8.209993E-01	-8.214553E-01
	Mean	-7.906871E-01	-7.955420E-01	-7.850579E-01	**-8.128026E-01**	-7.895534E-01	-8.024779E-01
	Median	-8.051968E-01	-8.035556E-01	-8.021446E-01	**-8.167530E-01**	-8.069657E-01	-8.109307E-01
	Std	0.0394106478	0.0239844032	0.0359429019	0.010401451	0.036606682	0.0224466185
	FR	1	1	1	1	1	1
C02	Best	-2.028941E+00	-2.070596E+00	-1.937690E+00	-1.962567E+00	-2.104664E+00	**-2.129791E+00**
	Mean	-1.501616E+00	-1.316971E+00	-1.524262E+00	-1.612921E+00	-1.579260E+00	**-1.658160E+00**
	Median	-1.649695E+00	-1.344832E+00	-1.592617E+00	-1.679900E+00	-1.868809E+00	**-1.889308E+00**
	Std	0.4598407083	0.5214896663	0.3730499485	0.3252846022	0.9712334592	0.9770352424
	FR	0.92	0.92	0.96	1	1	1
C03	Best	1.798732E-04	2.147231E-04	4.299019E-04	1.945596E-04	2.293741E-04	**1.213162E-04**
	Mean	1.606439E+01	1.835875E+01	6.718584E+01	6.695065E-03	**5.232345E-03**	9.963366E+08
	Median	2.867647E+01	2.867633E+01	2.867651E+01	1.452650E-03	**1.106168E-03**	1.859524E-03
	Std	14.5257983188	14.047622078	228.6963297386	0.0143275485	0.009571255	4881032690.67571
	FR	1	1	1	0.96	1	0.96
C04	Best	2.630825E-03	**1.627246E-03**	2.834130E-03	2.221984E-03	2.635186E-03	2.809565E-03
	Mean	4.292232E-03	6.940081E-02	5.389754E-03	8.507516E-03	8.755445E-03	**4.158587E-03**
	Median	4.670455E-03	3.972638E-03	4.088333E-03	4.924997E-03	4.230594E-03	**3.701366E-03**
	Std	0.0010013885	0.2050916414	0.0044313899	0.0137383827	0.0153783007	0.0015296364
	FR	0.24	0.4	0.52	0.52	0.64	0.4
C05	Best	-4.286187E+02	-4.445453E+02	-4.503970E+02	**-4.683370E+02**	-4.508990E+02	-4.614857E+02
	Mean	-2.831345E+02	-3.583624E+02	-3.170837E+02	-4.270805E+02	-4.065930E+02	**-4.331868E+02**
	Median	-2.406986E+02	-4.185614E+02	-3.254586E+02	**-4.436825E+02**	-4.257360E+02	-4.407806E+02
	Std	119.9259529885	99.1809643142	105.1800897502	59.964435771	50.9423882535	24.8358735812
	FR	0.4	0.52	0.4	0.96	0.96	1
C06	Best	-5.074999E+02	-5.257675E+02	-5.216241E+02	-5.251707E+02	-5.252116E+02	**-5.260180E+02**
	Mean	-3.172666E+02	-3.690244E+02	-3.356645E+02	-5.190846E+02	-5.177702E+02	**-5.210443E+02**
	Median	-3.014466E+02	-3.576268E+02	-2.889154E+02	-5.201676E+02	-5.176497E+02	**-5.208440E+02**
	Std	141.0614944537	120.9432610769	134.4011410656	3.6725279181	4.2636813877	2.5851008628
	FR	0.6	0.52	0.56	0.96	0.96	1
C07	Best	2.452761E+06	1.880186E+06	8.581103E+05	3.470941E+05	6.998194E+05	**3.077415E+05**
	Mean	2.121972E+07	1.649039E+07	2.825534E+07	1.539439E+07	**1.177201E+07**	2.092193E+07
	Median	1.578305E+07	1.013803E+07	1.342904E+07	1.054680E+07	**9.856244E+06**	1.209597E+07
	Std	20463490.6624698	14416514.8359341	34220955.4556132	22314350.035015	9327645.68603846	28078234.0290638
	FR	1	1	1	1	1	1
C08	Best	1.297202E+06	1.599710E+06	**9.695534E+05**	4.263515E+06	1.056525E+06	2.083105E+06
	Mean	2.424638E+07	2.361493E+07	3.534283E+07	2.123878E+07	**1.865897E+07**	1.919292E+07
	Median	**9.620651E+06**	1.803990E+07	2.188032E+07	1.241214E+07	1.132263E+07	1.873023E+07
	Std	28942746.0678157	21501185.3510753	43119161.1823233	29931033.7193782	32390226.9499147	13244617.0111663
	FR	1	1	1	1	1	1
C09	Best	1.170386E+08	2.000426E+08	1.356771E+08	**5.725403E+07**	7.028435E+07	2.096152E+08
	Mean	2.054205E+09	3.850938E+09	1.645751E+09	1.327185E+09	**1.086642E+09**	1.288078E+09
	Median	1.493080E+09	1.985780E+09	1.324588E+09	9.389505E+08	**8.719467E+08**	1.032037E+09
	Std	1928688451.25335	4048722194.13522	1410028055.98694	1267497608.02485	848643583.035502	974522816.02476
	FR	1	1	1	1	1	1
C10	Best	1.132249E+08	1.634377E+08	**5.290046E+07**	2.218009E+08	2.565191E+08	8.880810E+07
	Mean	2.232365E+09	2.671565E+09	2.377886E+09	**1.496530E+09**	1.810777E+09	1.664864E+09
	Median	1.348975E+09	1.717629E+09	1.541441E+09	1.236857E+09	1.173705E+09	**8.029145E+08**
	Std	2142461081.57498	2781160903.83436	2160440276.17879	1082746865.09996	1463606591.87325	1842986859.50566
	FR	1	1	1	1	1	1
C11	Best	-	-	-	**-3.387906E-04**	-3.236919E-04	-3.321988E-04
	Mean	-	-	-	**-3.163634E-04**	-2.722019E-04	-2.953903E-04
	Median	-	-	-	**-3.162091E-04**	-3.112411E-04	-2.931030E-04
	Std	-	-	-	1.39122048953607E-005	8.20391357873681E-005	3.44237647535907E-005
	FR	0	0	0	0.4	0.24	0.28
C12	Best	-1.982239E-01	-1.979244E-01	-1.980289E-01	**-1.985701E-01**	-1.983240E-01	-1.983693E-01
	Mean	6.489327E-02	-5.342994E-02	-7.774490E-02	**-1.007977E-01**	3.070812E-02	6.206711E-02
	Median	-1.141201E-01	-9.150999E-02	**-1.798003E-01**	-1.338328E-01	-6.292011E-02	-8.554155E-02
	Std	0.4825476505	0.1498712645	0.2411392218	0.1095566738	0.4151098755	0.4111699527
	FR	0.72	0.84	0.72	0.6	0.84	0.52
C13	Best	-4.339204E+01	-4.387888E+01	-4.631067E+01	-4.073612E+01	**-5.846836E+01**	-4.115957E+01
	Mean	-3.614333E+01	-3.597381E+01	-3.623899E+01	**-3.708869E+01**	-3.663108E+01	-3.616252E+01
	Median	-3.576710E+01	-3.582029E+01	-3.567818E+01	**-3.716624E+01**	-3.542507E+01	-3.570400E+01
	Std	2.9591297483	2.4989848671	3.4593544343	2.2505044482	5.2314679339	2.7788867376
	FR	1	1	1	1	1	1
C14	Best	3.095757E+09	3.254461E+09	1.588369E+09	3.114444E+09	2.501509E+09	**1.463029E+09**
	Mean	3.558607E+10	2.699306E+10	2.147086E+10	**2.019372E+10**	3.296048E+10	3.741909E+10
	Median	2.073490E+10	2.095488E+10	**1.255382E+10**	1.675086E+10	2.076095E+10	2.808929E+10
	Std	44049821653.3391	21147617118.3107	25235521446.6724	11972785993.6595	38515200922.5151	32332567429.2116
	FR	1	1	1	1	1	1
C15	Best	2.160675E+01	2.160935E+01	2.160803E+01	**4.780581E-04**	1.398412E-03	1.428983E-03
	Mean	2.296881E+01	2.376645E+01	2.344497E+01	2.019449E+01	1.573819E+06	**1.845420E+01**
	Median	2.238216E+01	2.262685E+01	2.280737E+01	2.187170E+01	2.250617E+01	**2.173677E+01**
	Std	1.8098464166	2.3593478916	1.8224038917	12.5665406753	7868988.16330518	9.2516359117
	FR	1	1	1	1	1	1
C16	Best	1.086953E-10	2.547318E-10	1.465367E-10	1.298822E-10	1.341258E-10	**5.059697E-11**
	Mean	7.671159E-10	9.239124E-07	4.129572E-09	9.750248E-09	2.614199E-08	**6.452409E-10**
	Median	4.151449E-10	6.906006E-10	5.473482E-10	5.305633E-10	5.426494E-10	**3.288210E-10**
	Std	1.13953724814516E-009	0.000004608	1.55011233825527E-008	4.14527039896093E-008	1.25597387814057E-007	6.90669799647074E-010
	FR	1	1	1	1	1	1
C17	Best	1.127200E-04	5.289437E-05	8.640277E-05	2.992582E-05	4.223067E-05	**1.261989E-05**
	Mean	3.918280E-03	3.319975E-03	3.489063E-01	7.862078E-04	2.711514E-04	**2.283398E-04**
	Median	6.118177E-04	1.836753E-04	3.017479E-04	6.823881E-04	1.950510E-04	**1.826438E-04**
	Std	0.0081271666	0.0128408545	1.4811419138	0.00052056	0.000295821	0.0002068771
	FR	1	1	1	1	1	1
C18	Best	2.176073E-11	9.380175E-11	4.063920E-11	1.601786E-11	1.143439E-11	**6.581079E-12**
	Mean	7.989147E-03	9.103697E-03	1.728619E-04	4.437594E-03	5.233829E-04	**2.429586E-05**
	Median	8.576027E-10	2.669062E-09	7.960236E-10	9.299224E-10	2.927154E-10	**2.260649E-10**
	Std	0.0268763837	0.0376588207	0.0008619611	0.0178380336	0.0026168765	0.0001105333
	FR	1	1	1	1	1	1

Chapter 6

A Multimeme Differential Evolution Framework

This chapter presents a multimeme coordination scheme based on the cost-benefit of local search operators, derived from the three studies presented in the previous chapter.

6.1 Cost-Benefit Multimeme Differential Evolution

The proposed approach, named Cost-Benefit Multimeme Differential Evolution (CoBe-MmDE), is aimed to coordinate a pool of local search operators (LSOs) within a Differential Evolution (DE) algorithm under constrained search spaces. The proposed method is based on a meta-Lamarckian logic [143] by using a cost-benefit scheme to control the LSO selection. Unlike the cost-benefit framework introduced in [144], the "benefit" (performance) of every LSO is computed by a different measure in this work, which is based on the success rate accumulated during every LSO activation. On the other hand, the adaptation in this proposal deals with a particular philosophy, considering the type of problem being tackled and the global search algorithm analyzed in this work.

6.1.1 Cost-Benefit Local Search Coordination

The LSO coordination mechanism, the main contribution of this study, works under a probabilistic rule, which is calculated using Equation 6.1, and it is composed of two main elements. The first element is the fitness evaluation rate. Such value is considered because, the complexity of the algorithms can be measured by the number of fitness function evaluations, and there are LSOs which perform few iterations, but the fitness function is computed several times. Therefore, the fitness function evaluations (*FEs*) carried out by the LSOs can be seen as a cost of exploitation. The second element is the LSO benefit. In this work, the success rate is considered to measure this LSO property. The adaptive probability β is defined as follows:

$$\beta = \left(\frac{lsoFE_t}{\sum_{i=1}^{t} lsoFE_i} \right) \times \left(\frac{lsoTS}{t} \right) \tag{6.1}$$

where, $lsoFE_t$ is the current number of fitness function evaluations of a given LSO, while $\sum_{i=1}^{t} lsoFE_i$ is the accumulated number of fitness evaluations during all LSO's

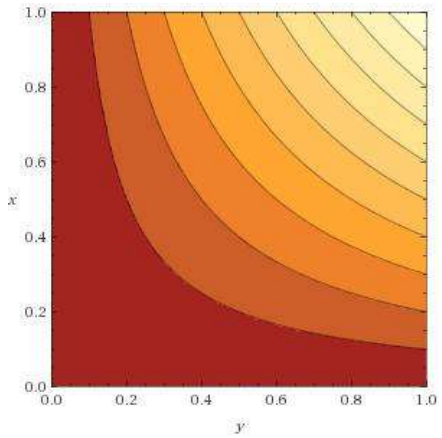


FIGURE 6.1: Contour plot of proposed probability β. x-axis expresses the values that can take the first element of the measure (*fitness evaluation rate*). y-axis expresses the values that can take the second element of the measure (*success rate*)

applications (t), and finally $lsoTS$ is the sum of those times where the LSO obtained a successful solution vector, i.e., when the new vector is better than the initial vector.

Every LSO has associated a β value, which varies between 0 and 1. The β value is updated after each time the LSO is applied, in order to updated the LSO activation probability for the next generation G as well. Therefore at the algorithm beginning, it is necessary to prefix an initial value greater than zero for the corresponding β associated to each LSO.

The expected behavior of β is to obtain values ≈ 1 in the first applications of the LSO and decrease it with each LSO usage, until values ≈ 0 are reached. Figure 6.1 shows the possible values that the proposed measure β can take, depending on the cost (fitness evaluation rate) and benefit (success rate). However, cases can occur in which the first use of one of the LSOs does not obtain a successful solution, following Equation 6.1, that LSO would have zero chance of being reused. This behavior prevents the use of an inadequate LSO for the problem being solved. On the other hand, the decrement speed of the the probability of use of each LSO decreases, is based on the number of fitness evaluations spent by each one of them. That implies that a proper configuration for the stop criterion within a given LSO could allow maintaining a relatively high probability to be activated.

The exploitation area for each LSO is a random selection process, which consists of two steps: (1) population sorting by using the ε comparison (see Equation 3.11), and (2) randomly selecting a vector of the ordered population, considering the first ten percent of the population, ensuring to exploit promising areas of the search space.

Finally, the Lamarckian learning [63] is considered as a replacement mechanism, i.e., the solution generated by each LSO is always kept for the next generation. The algorithm selects the worst vector of the population to be replaced by new vector generated by the LSO.

Although the proposed mechanism is capable of coordinating several LSOs, in this study three well-known direct search methods are implemented, Hooke-Jeeves (HJ) [37] , Nelder-Mead (NM) [38], and Hill Climbing (HC) [40]. Every LSO uses a maximum allowable fitness evaluation budget for each application. All three methods have been shown to have a competitive performance in constrained problems

[19, 145]. Whereas HJ, through its exploratory and pattern movements, obtains search directions approaching the gradient subspace, HC provides randomness in its exploration process, and NM can generate contraction and expansion movements in the search space through the initial simplex.

The complete process of CoBe-MmDE is described in Algorithm 20. Highlighted lines show the algorithmic coordination, which is applied after the DE replacement procedure.

6.2 Experiments and Results

To validate the efficiency of the proposed CoBe-MmDE, a set of experiments has been carried out on eighteen well-known benchmark problems [132] with different search space dimensionality, 10 and 30 dimensions. The characteristics of the problems are summarized in Table B.1.

The experiments are divided into three stages as follows: The first experiment is aimed to analyze the behavior of the local search coordination mechanism. For this purpose, the rate of LSO usage, the percentage of LSO fitness evaluations and the success ratio of LSO for each test problem are computed. Afterwards, the results are compared against three MmDE versions, which implement Random, Frequency (coordination proposed in [29]), and population Diversity-based local search coordination mechanism (proposed in [28]), respectively. The second experiment measures the performance of CoBe-MmDE through the feasibility rate [132], the success rate, and success performance [135]. Moreover, CoBe-MmDE is compared against four competitive DE variants for numerical optimization: EPSDE-LS [29], FDSADE [146], MADE [147], MS-CAP [148]. The ε-constrained method was implemented as constraint handler for each algorithm. Finally, the third expriment assesses the CoBe-MmDE numerical results by comparing them against those of state-of-the-art algorithms experiment for nature inspired constrained optimization, such as E-MODE [98], SAMO-DE [149], ε-DEag [14], DE-DPS [150] and, ECHT-ARMOR-DE [151].

For each experiment, twenty-five independent runs are computed. The parameter values used in the experiments are described in Section 6.2.1.

6.2.1 Parameter Setting

The parameter setting was conducted in two phases: In the first phase global and local search parameters were fine-tuned separately by using the IRACE tool [136] with a set of representative test problems as a training set for the tuning process. The parameters suggested in [14] and [5] were used as the starting point for such process. The second phase adjusted the parameters obtained from the first phase within the complete multimeme differential evolution proposed (CoBe-MmDE), by using a random selected set of test functions. This last phase was inspired by the method proposed in [152]. The obtained parameter values and adopted in the experiments are shown in Table 6.1.

Regarding the MmDE versions with different local search coordination mechanism (Rand-MmDE, Freq-MmDE, SF-MmDE), they used the same parameters that CoBe-MmDE for DE and LSOs. However, Freq-MmDE and SF-MmDE require some extra

Algorithm 20 CoBe-MmDE

Randomly generate an initial population of vectors $P_0 = (X_{0,i}, \ldots, X_{0,Pmax})$
Calculate the fitness of each vector in the initial population.
Set the ε value using Equation 3.13
Set $\beta_{hc} \leftarrow 0.1$, $\beta_{hj} \leftarrow 0.1$, and $\beta_{nm} \leftarrow 0.1$ {initial values of the LSO activation probabilities}
repeat
  for $i \leftarrow 1, Pmax$ **do**
    Randomly select $r0$,$r1$,$r2 \in [1, Pmax]$ and $r0 \neq r1 \neq r2 \neq i$
    Randomly select $J_{rand} \in [1, D]$
    for $j \leftarrow 1, D$ **do**
      if $rand_j \leq Cr$ Or $j = Jrand$ **then**
        $u_{G,i,j} = x_{G,r0,j} + F(x_{G,r1,j} - x_{G,r2,j})$
      else
        $u_{G,i,j} = x_{G,i,j}$
      end if
    end for
    if $U_{G,i} \leq_{\varepsilon} X_{G,i}$ using Equation 3.11 **then**
      $X_{G+1,i} = U_{G,i}$
    else
      $X_{G+1,i} = X_{G,i}$
    end if
  end for
  Sort Population P_{G+1} using Equation 3.11
  if $\beta_{hc} <$ rand(0, 1) **then**
    Randomly select $r \in [1, (Pmax * 0.1)]$
    Set $X_{new} \leftarrow$ **Hill Climbing**$(X_{G,r})$
    Set $X_{G,worst} \leftarrow X_{new}$
    Update β_{hc} using Equation 6.1
  end if
  if $\beta_{hj} <$ rand(0, 1) **then**
    Randomly select $r \in [1, (Pmax * 0.1)]$
    Set $X_{new} \leftarrow$ **Hooke-Jeeves**$(X_{G,r})$
    Set $X_{G,worst} \leftarrow X_{new}$
    Update β_{hj} using Equation 6.1
  end if
  if $\beta_{nm} <$ rand(0, 1) **then**
    Randomly select $r \in [1, (Pmax * 0.1)]$
    Set $X_{new} \leftarrow$ **Nelder-Mead**$(X_{G,r})$
    Set $X_{G,worst} \leftarrow X_{new}$
    Update β_{nm} using Equation 6.1
  end if
  Update ε value using Equation 3.12
  $G = G + 1$
until *MaxFEs* is reached

TABLE 6.1: Parameters values for Differential Evolution (DE), local search operators (Hooke-Jeeves (HJ), Hill Climbing (HC) and Nelder-Mead (NM)) and the constraint handler (ε-Constrained)

Algorithm	**Parameter**	**Value**	
		10 D	30 D
DE	*MaxFEs*	$2.0E+05$	$6.0E+05$
	Pmax	80	
	Cr	0.9	
	F	0.55	
HJ	α	rand(2, 3)	
	δ	rand(0.75, 0.9)	
	MaxFEs	rand(500,900)	
HC	σ	rand(0,1)	
	MaxFEs	rand(500,900)	
NM	β	rand(0.7, 0.9)	
	γ	rand(3, 4)	
	MaxFEs	rand(500,900)	
ε-Constrained	θ	0.75	
	cp	9.5	
	Gc	1100	

parameter for LSO coordinations which were obtained by analyzing each LSO performance for a set of test functions inspired in [145]:

- Freq-MmDE uses the frequency of LSO activation F_{LS} = 30, that means if (t mod F_{LS}) = 0 a LSO is randomly selected from the pool [29].
- SF-MmDE requires four parameter (α, β, a, b) for each LSO in the pool, since the Beta distribution is applied to perform the probability 4.5 of LSO activation [28]: for HJ = (2, 5, 0, 0.2), regarding HC = (2, 2, 0.1, 0.3) and NM = (2, 2, 0.2, 0.6).

On the other hand, since the implemented algorithms (EPSDE-LS [29], FDSADE [146], MADE [147], and MS-CAP [148]) were adapted for CNOPs, using the epsilon-constrained method, the parameter values required for each approach were adapted similarly as previously described, based on the parameters suggested by the authors. Those are outlined below:

- Ensemble of Parameters and Strategies Differential Evolution empowered by Local Search (EPSDE-LS): $Pmax$ = 50, frequency of LSO activation F_{LS} = 100, budget condition B_{LS} = rand(500,900). Parameter pools were chosen as P_{CR} = { 1.0, 0.9, 0.8 }, P_F = { 0.55, 0.65, 0.75 }. Pools of strategies, P_{cross} = {bin, exp} and P_{mut} = {cur-to-pbest/1, cur-to-rand/1}. LSOs parameters values used where those reported in [29].
- Fitness Diversity Self-adaptive Differential Evolution (FDSADE): Regarding population size, S^f_{pop} = 10, S^v_{pop} = 40. F_l = 0.1, F_u = 0.9 and K = 0.3 as suggested in [146].
- Multicriteria Adaptive Differential Evolution (MADE): $Pmax$ = 50, β = 0.7, p_{min} = 0.02, LP = 50 according to [147].
- Multi-Strategy Coevolving Aging Particles (MS-CAP): $Pmax$ = 20, L = 3, ε = 10^{-6} as reported in [148].

6.2.2 Results of the coordination mechanism behavior

The rate of LSO usage was computed by means of the average of local search activations of twenty-five runs for the eighteen test problems in 10D and 30D. Table 6.2 shows significant differences among the number of LSOs' activations by using different local search coordination mechanisms: Cost-Benefit (CoBe), Random (Rand), Frequency (Freq) [29], and based on the fitness diversity (SF) [28]. Those results also reveal that CoBe activates HJ, NM, and HC moderately in all test functions at 10D and 30D. On the other hand, Rand and Freq show a uniform distribution in the number of applications of the three LSOs. However, the Freq application rate parameter helps the algorithm not to make as much use of the LSO pool. Finally, the SF coordination mechanism, adapts the LSO use according to the fitness diversity, so that the average of the LSOs activations contrasts significantly in most problems. Therefore, the coordination mechanism based on diversity (SF) is susceptible to a prior analysis of each LSO to adopt the required parameters that modulate the probability of activating them during the MmDE evolutionary process.

On the other hand, the average percentage of fitness evaluations shown in Figures 6.2 and 6.3, suggests that the greatest computational effort yields on DE using CoBe. Because of the probability β proposed in this work (see Equation 6.1), which modulates the number of LSO activations, considers the number of evaluations spent by each LSO. Thus, the probability β decreases in LSOs with high success rate but a significant percentage of evaluations, leaving the global searcher the most of the computational effort. Regarding Freq, the user-defined parameter to control de LSO activations, allows moderating the excessive use of them. On the other hand, SF depends on the population diversity and leading to the fact that in some functions (C02, C05, C06, C12, and C13), SF executes much of the computational effort on the part of the LSOs. Finally, Rand performs significant fitness evaluations by the LSOs, of the estimated budget for the search process in all test problems.

Finally for this first experiment, Figures 6.4 and 6.5 show the average of success ratio (SR) values for each test problem at 10D and 30D. SR is performed by dividing the number of LSO success activation by the total of local search activation and determines the performance of a particular LSO. A successful activation is when the solution generated by the LSO improves its initial solution. While SR values close to one indicate a high LSO performance, those SR values close to zero show a low LSO performance. Results denote a higher perfomance by using SF in most test problems. However, CoBe shows a competitive performance in the LSO coordination. While mechanisms based on random coordination (Rand and Freq), show a lower perfomance of LSOs usage. Considering the fitness evaluations average consumed by each coordination mechanism and the local search activations, CoBe presents a balanced and competitive behavior in the coordination of LSOs within the evolutionary cycle of a MDE for CNOPs.

Regarding the previous results, CoBe is able to modulate the use of HC, HJ, and NM considering the computational effort and the success rate of every local search method. Also, it can be noticed that a particular LSO is not suitable for the exploitation in all test problems. For that reason, the importance of considering others LSOs that can compensate the capacity to exploit complex search spaces is remarked.

TABLE 6.2: Average of LSO activation during the MmDE evolution using different local search coordinators per test functions.

Problem	Algorithm	10D				30D			
		HJ	**HC**	**NM**	**Total**	**HJ**	**HC**	**NM**	**Total**
C01	CoBe	22.0	28.5	37.1	87.6	25.0	40.3	57.3	122.6
	Rand	159.5	159.9	160.1	479.4	193.3	195.2	198.9	587.5
	Freq	17.3	21.1	18.5	57.0	20.4	24.3	23.7	68.4
	SF	60.6	28.1	61.9	150.5	91.0	134.1	233.8	458.9
C02	CoBe	47.4	21.5	34.5	103.4	78.7	37.6	64.8	181.1
	Rand	140.6	143.9	142.3	426.7	319.2	315.5	313.9	948.5
	Freq	18.7	19.8	18.1	56.6	53.7	52.7	54.9	161.3
	SF	283.9	44.2	15.5	343.5	221.2	384.6	764.4	1,370.2
C03	CoBe	43.2	18.6	28.9	90.7	57.1	40.5	42.1	139.7
	Rand	141.3	139.1	143.0	423.5	179.8	173.3	177.1	530.2
	Freq	20.1	18.3	17.5	55.9	23.4	21.4	22.9	67.7
	SF	16.2	3.3	2.6	22.1	84.8	16.6	11.4	112.8
C04	CoBe	48.1	24.9	38.9	111.9	60.0	36.6	56.1	152.7
	Rand	124.0	126.3	127.7	378.0	177.7	186.5	180.1	544.3
	Freq	18.9	19.6	17.5	55.9	22.9	23.0	21.9	67.9
	SF	236.1	64.3	43.8	344.2	68.8	608.0	171.0	847.8
C05	CoBe	44.3	32.3	40.0	116.6	84.2	52.7	81.2	218.1
	Rand	125.7	131.1	122.6	379.4	312.9	303.8	305.3	921.9
	Freq	17.0	20.2	19.1	56.3	55.2	51.5	54.1	160.9
	SF	240.5	16.4	14.3	271.1	465.6	83.3	146.6	695.5
C06	CoBe	39.4	26.3	30.0	95.7	84.6	49.2	78.5	212.3
	Rand	149.5	149.5	146.2	445.2	310.2	304.7	298.0	912.9
	Freq	18.9	20.0	17.7	56.5	53.8	53.3	53.7	160.7
	SF	118.1	9.9	14.3	142.3	521.1	65.3	78.7	665.1
C07	CoBe	42.6	17.9	30.3	90.8	75.3	42.6	64.7	182.7
	Rand	146.7	148.3	153.7	448.7	308.9	308.3	309.2	926.4
	Freq	18.7	19.3	18.2	56.3	52.6	53.8	53.9	160.3
	SF	12.1	5.0	4.9	22.1	27.1	13.1	19.0	59.2
C08	CoBe	41.6	20.3	34.1	96.1	76.5	44.1	70.7	191.4
	Rand	140.3	144.6	146.9	431.9	305.7	307.5	312.0	925.1
	Freq	19.5	18.6	18.3	56.4	52.6	54.8	53.5	160.9
	SF	13.8	5.1	3.9	22.8	38.3	12.1	18.5	68.9
C09	CoBe	43.0	19.7	29.4	92.1	76.6	46.9	57.7	181.1
	Rand	140.1	140.9	138.9	419.8	301.7	301.5	296.6	899.7
	Freq	19.2	18.1	18.8	56.1	54.4	51.3	54.4	160.1
	SF	14.9	4.4	4.8	24.1	41.4	14.0	16.3	71.7
C10	CoBe	40.3	19.3	29.3	88.9	78.0	46.1	58.7	182.7
	Rand	139.5	136.1	138.7	414.4	297.9	297.4	305.9	901.2
	Freq	19.8	20.0	16.5	56.3	51.9	55.1	53.3	160.3
	SF	13.9	4.7	4.6	23.3	36.8	13.5	16.2	66.5
C11	CoBe	41.7	20.6	27.8	90.1	69.4	42.3	65.6	177.3
	Rand	150.4	148.6	146.8	445.8	307.2	319.3	313.7	940.2
	Freq	17.9	20.1	18.5	56.5	52.9	53.0	54.8	160.7
	SF	85.5	9.3	6.3	101.0	178.1	15.2	20.4	213.7
C12	CoBe	41.6	18.8	28.0	88.4	58.5	40.4	41.9	140.8
	Rand	136.5	138.1	138.3	412.9	175.5	180.1	176.2	531.9
	Freq	17.3	19.9	19.1	56.3	24.1	20.5	23.1	67.7
	SF	141.7	31.7	21.0	194.5	308.0	78.3	62.8	449.1
C13	CoBe	41.9	34.9	38.8	115.5	57.5	53.5	54.2	165.1
	Rand	93.8	95.7	92.3	281.9	150.5	149.8	148.8	449.1
	Freq	16.5	19.1	18.5	54.2	22.0	20.1	25.0	67.1
	SF	16.7	21.7	346.0	384.3	11.9	7.3	531.6	550.8
C14	CoBe	43.3	19.9	28.4	91.6	80.5	43.9	64.1	188.5
	Rand	138.6	134.8	141.1	414.5	292.7	297.4	300.2	890.3
	Freq	18.2	18.5	19.5	56.2	54.8	54.3	50.6	159.7
	SF	15.3	4.4	5.5	25.2	37.3	12.8	17.3	67.5
C15	CoBe	41.1	19.9	29.4	90.4	57.7	41.2	41.1	139.9
	Rand	137.9	137.9	138.6	414.3	171.1	178.3	169.6	519.0
	Freq	18.5	18.7	19.0	56.2	22.5	22.0	23.2	67.7
	SF	14.2	4.3	4.8	23.3	84.4	26.9	33.9	145.2
C16	CoBe	43.1	32.1	29.7	104.9	55.5	36.3	46.1	137.9
	Rand	133.8	133.9	138.1	405.8	189.7	199.7	197.5	587.0
	Freq	17.4	18.5	19.9	55.8	21.6	25.0	21.9	68.5
	SF	76.0	39.9	39.2	155.1	141.9	25.7	64.5	232.1
C17	CoBe	45.5	24.8	25.6	95.9	58.3	31.5	41.7	131.5
	Rand	121.4	128.0	124.3	373.7	196.9	195.3	194.2	586.5
	Freq	18.0	18.8	19.1	55.9	22.5	22.3	23.4	68.2
	SF	26.5	5.5	6.0	37.9	115.5	15.3	19.4	150.1
C18	CoBe	33.4	23.1	28.7	85.2	42.9	33.9	42.5	119.3
	Rand	113.9	114.6	115.0	343.5	190.7	193.3	196.3	580.3
	Freq	18.3	18.6	18.9	55.7	23.1	22.8	22.2	68.1
	SF	29.9	5.5	5.1	40.5	112.1	16.7	22.9	151.7

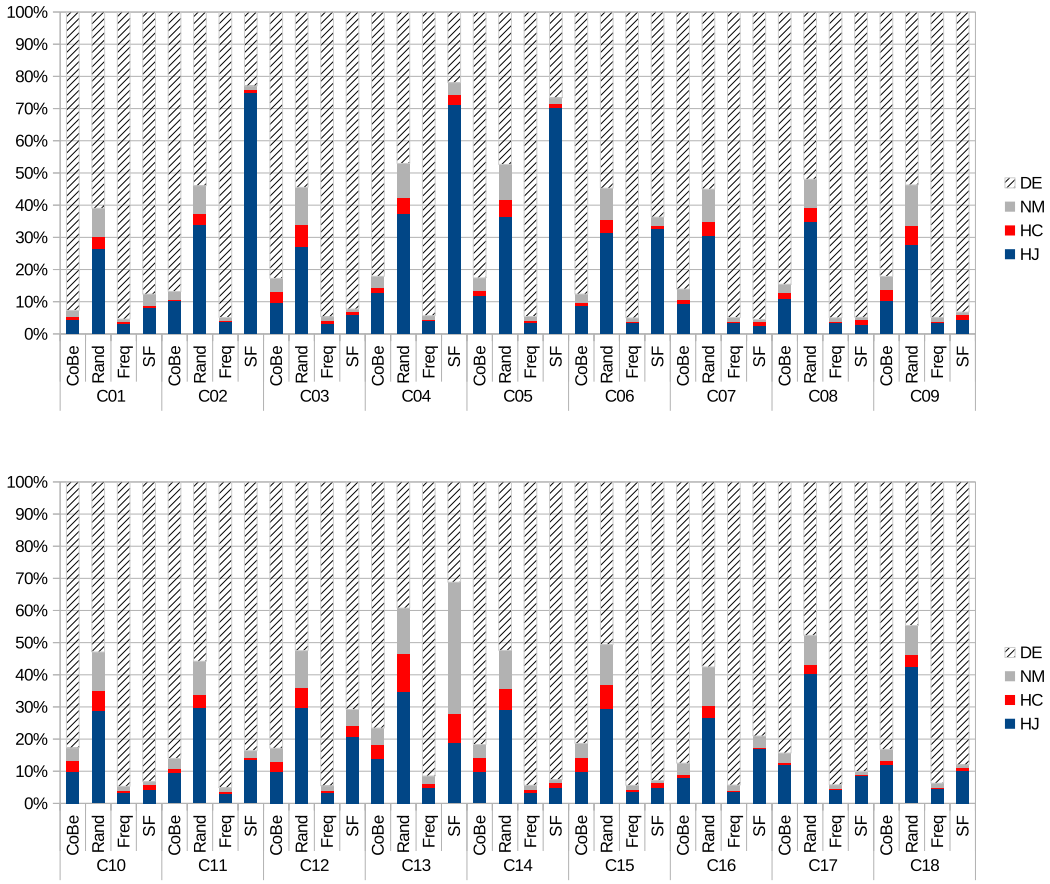


FIGURE 6.2: Average of fitness evaluations for global and local search operators for test problems at 10D, using different local search coordinators.

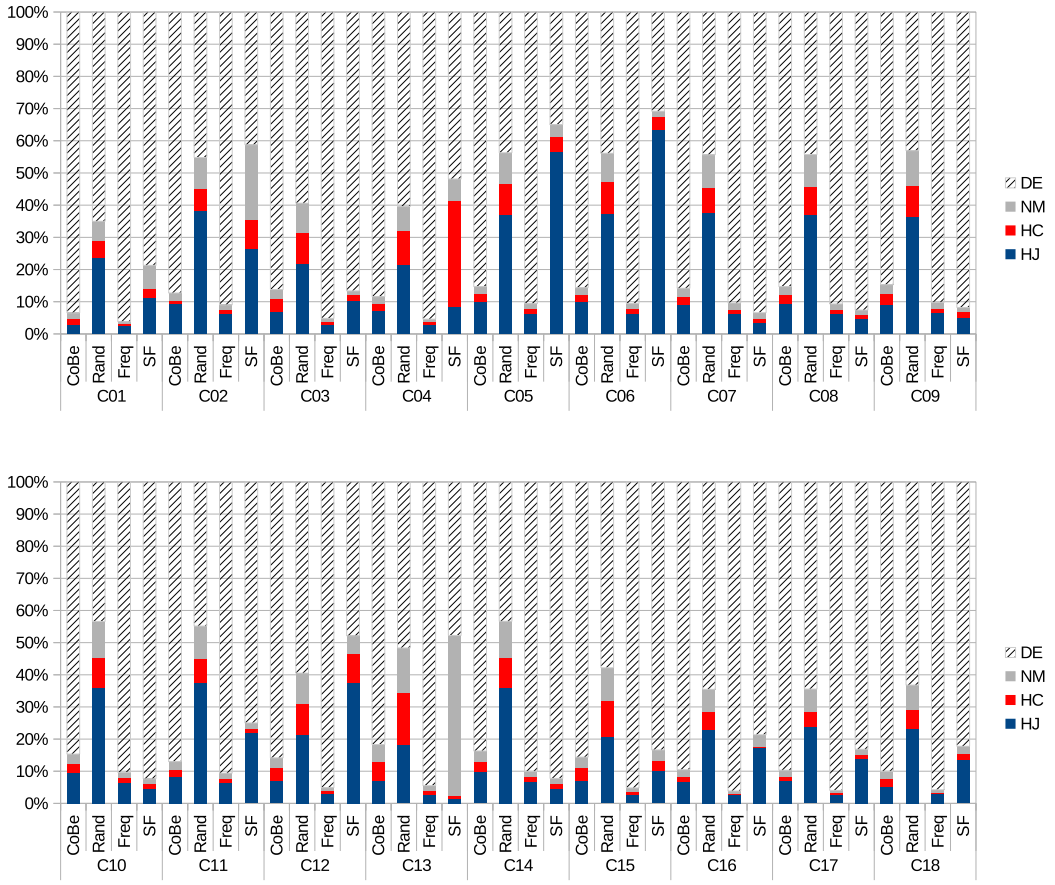


FIGURE 6.3: Average of fitness evaluations for global and local search operators for test problems at 30D, using different local search coordinators.

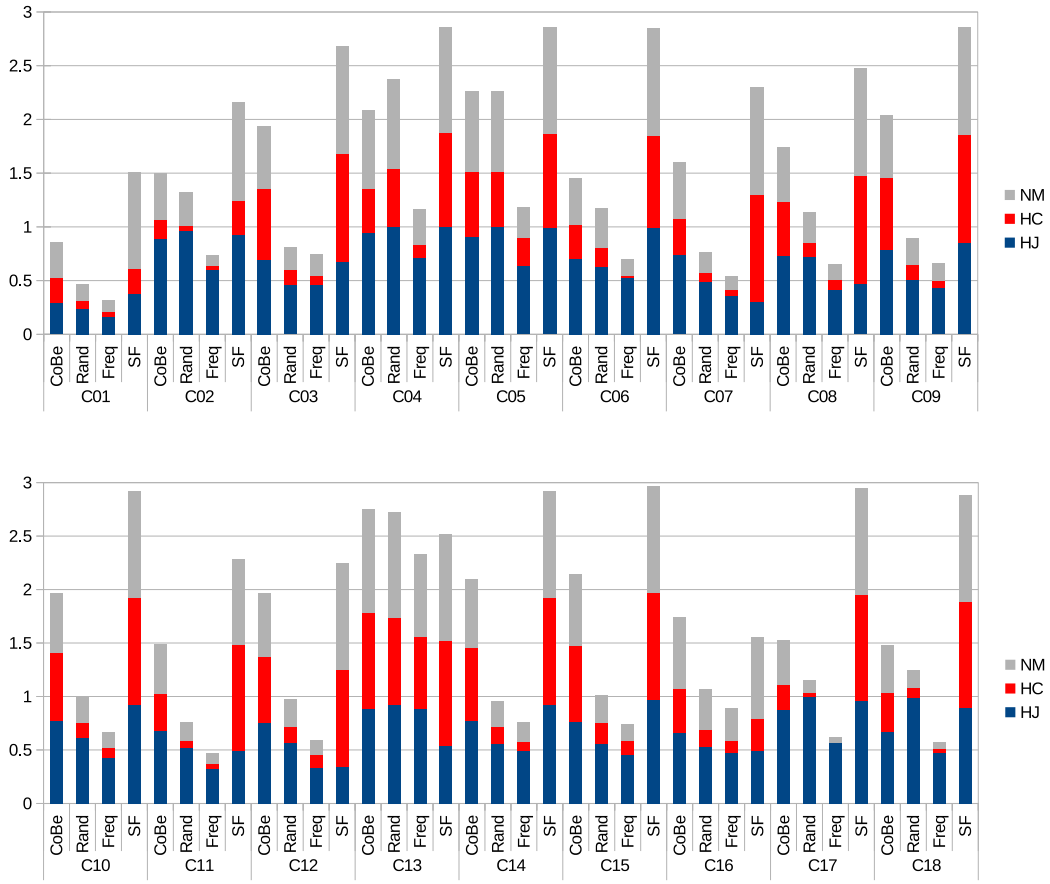


FIGURE 6.4: Success ratio for each LSO by using different local search coordinatiors for test problems at 10D.

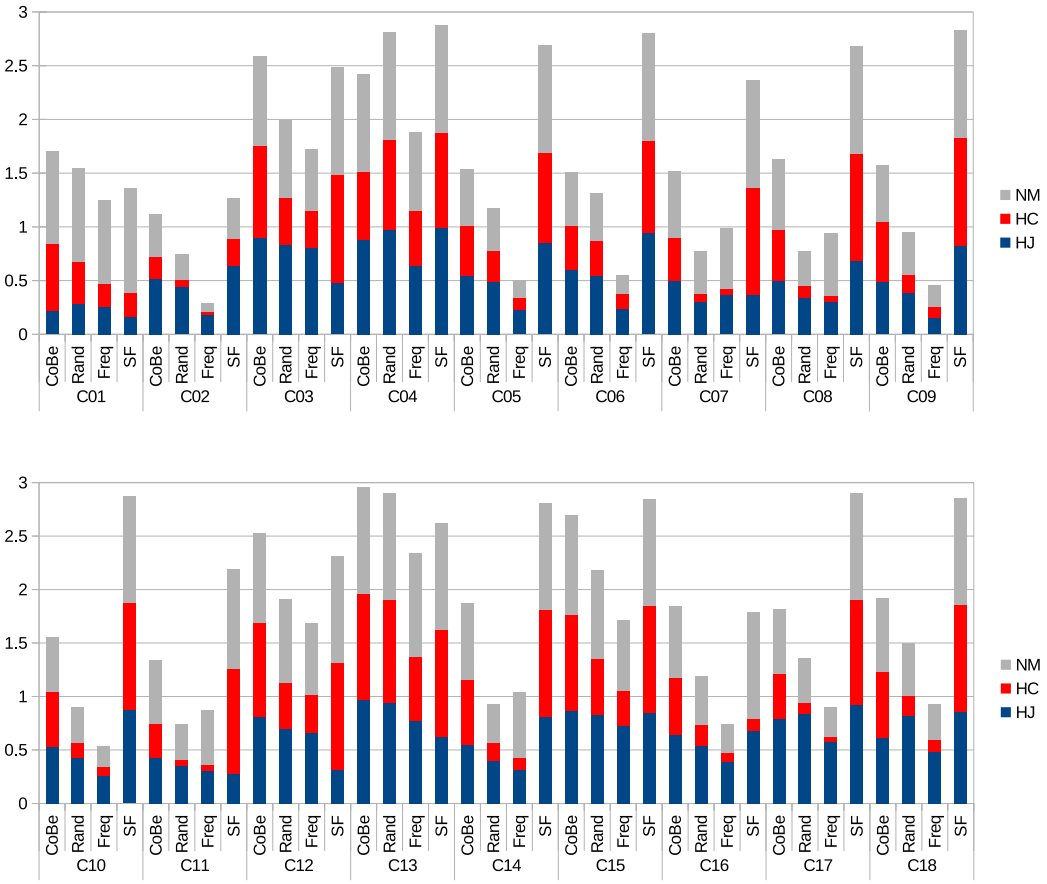


FIGURE 6.5: Success ratio for each LSO by using different local search coordinatiors for test problems at 30D.

6.2.3 Results of CoBe-MmDE performance

A feasible run is an execution of the algorithm where at least one feasible solution is found in *MaxFES*. A successful run is where the algorithm finds a feasible solution X satisfying $f(X) - f(X^*) \leq 0.0001$. $f(X^*)$ values, have been taken from the suplementary material reported in [98]. The performance measures considered for this analysis, feasibility rate (FR), and success performance (SP) are defined by Equations 6.2 and 6.3, respectively.

$$\textbf{FR} = \frac{\text{\# of feasible runs}}{\text{total runs}} \tag{6.2}$$

$$\textbf{SP} = \frac{\text{Mean(FES for successful runs)} \times \text{(\# of total runs)}}{\text{\# of successful runs}} \tag{6.3}$$

where Mean(FES for successful runs), is the mean of fitness evaluations that the algorithm requires to achieve a success run. Therefore, for this experiment the algorithm stops when the error value $(f(X) - f(X^*)) \leq 0.0001$ or the $MaxFES$ allowed are reached.

CoBe-MmDE performance results were compared against four competitive DE variants for numerical optimization: EPSDE-LS [29], FDSADE [146], MADE [147], MS-CAP [148]. The ε-constrained method was adopted as constraint handler for all implementations.

The results described in Tables 6.3 and 6.4 show a high feasibility rate (FR) of CoBe-MmDE. Considering 90% of the test problems (including 10D and 30D) CoBe-MmDE can find feasible solutions in all runs, except for C05, C06, C11, C12, and C17 test problems in 30D. In functions like C03 and C04 in 30D, the performance of CoBe-MmDE is higher to those of the other algorithms. Regarding the success performance (SP), that measures the average of fitness evaluations (FES) needed to achieve the best know fitness value $f(X*)$ for a test problem, CoBe-MmDE can obtain feasible solutions near or equal to the optimal in 100% test instances. Although EPSDE-LS, FDSADE, MADE and MS-CAP can find competitive SP values, CoBe-MmDE does not exceed the FES budget allowed in all test problems. Likewise, CoBe-MmDE shows a constant performance without being sensitive to the dimension of the test problems.

The error values per test problem are shown in Figures 6.6 and 6.7. Boxplots suggest robustness in most test problems. The results also show that CoBe-MmDE can generate successful runs in all instances.

The performance results suggest that the CoBe-MmDE can obtain feasible and competitive solutions for CNOPs. Although CoBe-MmDE uses a simple implementation coordination mechanism, it is able to achieve performance that competes with more sophisticated DE variants concerning the number of components required for its operation, such as EPSDE-LS, FDSADE, MADE and MS-CAP, which adopt mutation operator coordination mechanisms and also parameter adaptation.

6.2.4 Results of numerical comparison

Numerical results for 10D test problems are presented in Tables 6.5 and 6.6, whereas Tables 6.7 and 6.8 show results for 30D test problems. The results are also compared

TABLE 6.3: Fesibility Rate (FR) and Success Performance (SP) for C01 to C09 test problems at 10D and 30D. Boldface remarks those best values.

Problem	Algorithm	10D		30D	
		FR	SP	FR	SP
C01	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	EPSDE-LS	*1.00*	200,033.88	*1.00*	600,024.00
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MS-CAP	*1.00*	200,036.00	*1.00*	600,040.00
C02	CoBe-MmDE	*1.00*	200,053.88	*1.00*	*600,000.00*
	EPSDE-LS	0.96	208,379.90	*1.00*	600,028.00
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MS-CAP	0.68	294,167.82	*1.00*	600,033.60
C03	CoBe-MmDE	*1.00*	*200,000.00*	**1.00**	**600,000.00**
	EPSDE-LS	*1.00*	200,025.84	0.04	15,000,575.00
	FDSADE	*1.00*	*200,000.00*	0.84	714,285.71
	MADE	*1.00*	*200,000.00*	0.36	1,666,666.67
	MS-CAP	*1.00*	200,030.40	0.80	750,033.75
C04	CoBe-MmDE	*1.00*	**200,000.00**	**1.00**	**600,112.96**
	EPSDE-LS	*1.00*	200,039.68	0.44	1,363,985.74
	FDSADE	0.76	263,157.89	0.00	-
	MADE	0.96	208,333.33	0.00	-
	MS-CAP	0.84	238,143.99	0.00	-
C05	CoBe-MmDE	0.92	217,449.39	**0.92**	**652,234.64**
	EPSDE-LS	*1.00*	200,035.88	0.32	1,875,057.81
	FDSADE	0.96	208,333.33	0.04	15,000,000.00
	MADE	*1.00*	**200,000.00**	0.88	681,818.18
	MS-CAP	0.92	217,428.17	0.16	3,750,062.50
C06	CoBe-MmDE	*1.00*	*200,000.00*	0.92	652,231.66
	EPSDE-LS	0.96	208,368.71	0.24	2,500,154.17
	FDSADE	0.84	238,095.24	0.16	3,750,000.00
	MADE	*1.00*	*200,000.00*	**0.96**	**625,000.00**
	MS-CAP	*1.00*	200,034.40	0.44	1,363,719.01
C07	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	EPSDE-LS	*1.00*	200,023.72	*1.00*	600,018.72
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MS-CAP	*1.00*	200,030.40	*1.00*	600,032.00
C08	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	600,053.08
	EPSDE-LS	*1.00*	200,035.44	*1.00*	600,036.48
	FDSADE	*1.00*	*200,000.00*	*1.00*	600,000.00
	MADE	*1.00*	*200,000.00*	*1.00*	600,000.00
	MS-CAP	*1.00*	200,032.80	*1.00*	600,036.00
C09	CoBe-MmDE	*1.00*	200,051.64	*1.00*	600,039.40
	EPSDE-LS	*1.00*	200,037.20	0.92	652,201.98
	FDSADE	*1.00*	*200,000.00*	*1.00*	600,000.00
	MADE	*1.00*	*200,000.00*	*1.00*	600,000.00
	MS-CAP	*1.00*	200,028.00	*1.00*	600,021.60

TABLE 6.4: Fesibility Rate (FR) and Success Performance (SP) for C10 to C18 test problems at 10D and 30D. Boldface remarks those best values.

Problem	Algorithm	10D		30D	
		FR	SP	FR	SP
C10	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	600,000.00
	EPSDE-LS	*1.00*	200,032.08	*1.00*	600,056.76
	FDSADE	*1.00*	*200,000.00*	*1.00*	600,000.00
	MADE	*1.00*	*200,000.00*	*1.00*	600,000.00
	MS-CAP	*1.00*	200,030.40	*1.00*	600,036.80
C11	CoBe-MmDE	*1.00*	**200,000.00**	0.76	789,522.16
	EPSDE-LS	*1.00*	200,033.40	**1.00**	**600,068.60**
	FDSADE	0.96	208,333.33	0.12	5,000,000.00
	MADE	0.84	238,095.24	0.68	882,352.94
	MS-CAP	0.76	263,193.91	0.76	789,522.16
C12	CoBe-MmDE	*1.00*	*200,000.00*	0.60	1,000,219.89
	EPSDE-LS	*1.00*	200,025.08	**1.00**	**600,048.88**
	FDSADE	*1.00*	*200,000.00*	0.64	937,500.00
	MADE	0.96	208,333.33	0.76	789,473.68
	MS-CAP	*1.00*	200,026.40	0.92	652,208.88
C13	CoBe-MmDE	*1.00*	200,045.08	*1.00*	*600,000.00*
	EPSDE-LS	*1.00*	200,036.36	*1.00*	600,056.24
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MS-CAP	*1.00*	200,031.20	*1.00*	600,032.80
C14	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	600,050.68
	EPSDE-LS	*1.00*	200,025.40	*1.00*	600,060.44
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	0.96	208,333.33	0.96	625,000.00
	MS-CAP	1.00	200,024.00	*1.00*	600,034.40
C15	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	600,123.44
	EPSDE-LS	*1.00*	200,022.44	*1.00*	600,020.16
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MS-CAP	*1.00*	200,037.60	*1.00*	600,039.20
C16	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	600,121.84
	EPSDE-LS	*1.00*	200,022.12	*1.00*	600,016.04
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MS-CAP	0.96	208,372.40	*1.00*	600,034.40
C17	CoBe-MmDE	0.92	217,444.09	0.92	652,300.09
	EPSDE-LS	*1.00*	200,036.60	*1.00*	600,039.24
	FDSADE	0.92	217,391.30	*1.00*	*600,000.00*
	MADE	*1.00*	**200,000.00**	*1.00*	*600,000.00*
	MS-CAP	0.60	333,388.89	0.60	1,000,071.11
C18	CoBe-MmDE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	EPSDE-LS	*1.00*	200,034.00	*1.00*	600,057.84
	FDSADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MADE	*1.00*	*200,000.00*	*1.00*	*600,000.00*
	MS-CAP	*1.00*	200,031.20	*1.00*	600,034.40

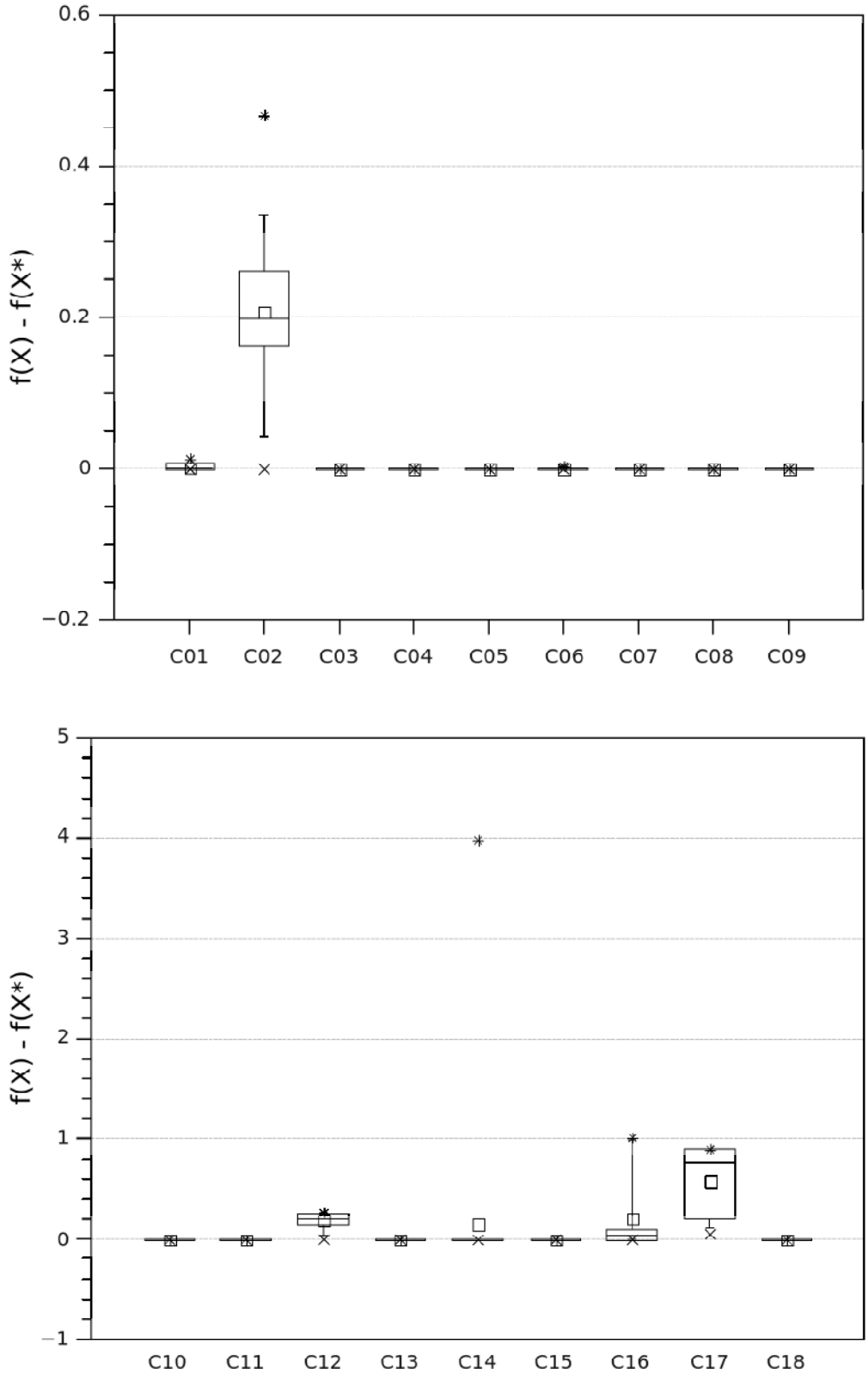


FIGURE 6.6: Boxplots of error values in 10D test problems.

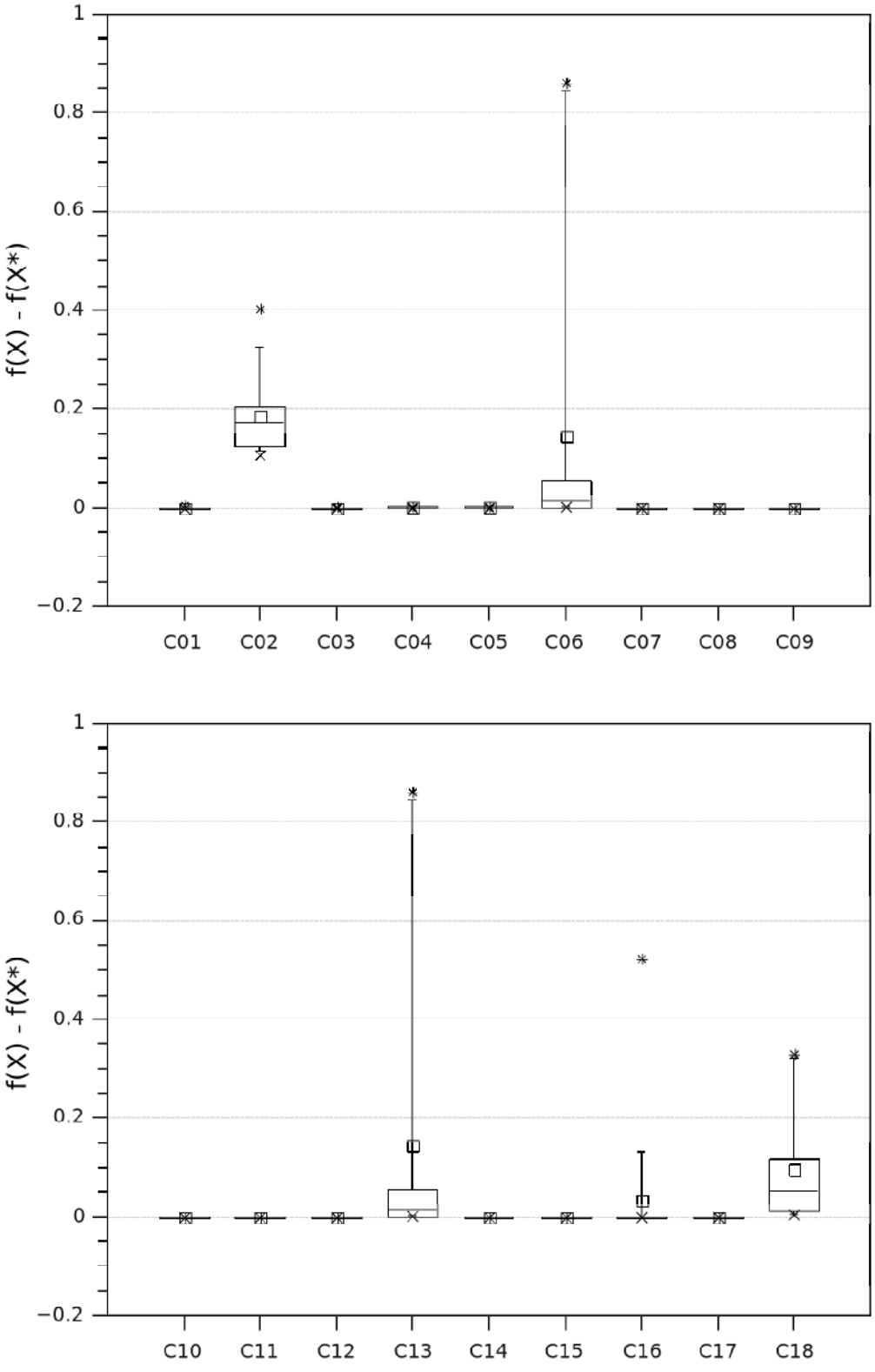


FIGURE 6.7: Boxplots of error values in 30D test problems.

against those of a group of state-of-the-art algorithms for constrained optimization: E-MODE [98], SAMO-DE [149], ε-DEag [14], DE-DPS [150] and, ECHT-ARMOR-DE [151]. All algorithms are based on Differential Evolution. Nevertheless, each

proposal uses different mechanisms to improve the DE search capabilities. While E-MODE and SAMO-DE adopt mechanisms to handle multiple DE operators, ε-DEag uses a gradient-based mutation as LSO with an archive. On the other hand, DE-DPS implements a mechanism to select the best performing combinations of parameters dynamically. Finally, ECHT-ARMOR-DE uses a ranking technique to select mutation operators. 95%-confidence Wilcoxon rank-sum-test was computed to determine the significant differences between algorithms. The comparison is summarized in Table 6.9. The approximation sign ("$\approx$") determines that there is no significant difference between the two algorithms, while sign ("$-$") means that the first algorithm is significantly worse.

The statistical results show that CoBe-MmDE is competitive regarding the compared state-of-the-art algorithms. Moreover, the proposed approach is simpler to implement than the other algorithms compared, since no second-order information of test functions are required as well as archives storage or calculations of diverse populations. On the other hand, the CoBe allows the control of multiple LSOs without requiring previous information from them. However, it is important to analyze the behaviors of each LSO, to adopt those that have a higher contribution to the exploitation of promising areas in constrained search spaces.

6.3 Conclusions

Numerical results demonstrated that the proposed method was suitable to coordinate a set of LSOs adequately within a memetic Differential Evolution scheme under constrained search spaces. According to the experimental results, the coordination scheme proposed (CoBe) can modulate the LSOs activations robustly and efficiently, showing stability of the behavior in different types of constrained problems. Likewise, CoBe obtained a competitive performance with three different types of coordination mechanisms. Finally, CoBe-MmDE proved to be competitive in both performance and numerical results compared to algorithms with more sophisticated operations. The results also demonstrated the versatility of DE to adopt memetic schemes, which leaves the question if CoBe can be adapted within multimeme schemes based on other types of evolutionary algorithms or swarm intelligence algorithms, and this is precisely the initial topic of future research.

TABLE 6.5: 10D statistical results comparison. Boldface remarks those best values and Italics remark those values that are equal than the other reported. Std denotes the standard deviation of 25 runs per test problem

Problem	Algorithm	Best	Mean	Std
C01	CoBe-MmDE	*-7.473104E-01*	-7.418165E-01	1.201706E-02
	E-MODE	*-7.473104E-01*	*-7.473104E-01*	2.451306E-16
	SAMO-DE	*-7.473104E-01*	-7.470402E-01	1.350638E-03
	εDEag	*-7.473104E-01*	-7.470402E-01	1.323339E-03
	DE-DPS	*-7.473104E-01*	*-7.473104E-01*	**2.266230E-16**
	ECHT-ARMOR-DE	-7.473000E-01	-7.470000E-01	1.400000E-03
C02	CoBe-MmDE	-2.169218E+00	-2.030492E+00	1.031954E-01
	E-MODE	*-2.277711E+00*	**-2.277711E+00**	**2.540214E-10**
	SAMO-DE	-2.277709E+00	-2.276842E+00	1.154957E-03
	εDEag	-2.277702E+00	-2.258870E+00	2.389779E-02
	DE-DPS	*-2.277711E+00*	-2.277512E+00	2.540350E-04
	ECHT-ARMOR-DE	-2.277700E+00	-2.277000E+00	3.300000E-03
C03	CoBe-MmDE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	E-MODE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	SAMO-DE	*0.000000E+00*	4.173000E-23	1.640510E-22
	εDEag	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	DE-DPS	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	ECHT-ARMOR-DE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
C04	CoBe-MmDE	-9.996233E-06	-9.939055E-06	6.820370E-08
	E-MODE	*-1.000000E-05*	*-1.000000E-05*	*0.000000E+00*
	SAMO-DE	*-1.000000E-05*	*-1.000000E-05*	1.446083E-11
	εDEag	-9.992345E-06	-9.918452E-06	1.546730E-07
	DE-DPS	*-1.000000E-05*	*-1.000000E-05*	9.096330E-15
	ECHT-ARMOR-DE	*-1.000000E-05*	*-1.000000E-05*	*0.000000E+00*
C05	CoBe-MmDE	**-4.836106E+02**	**-4.836106E+02**	1.292164E-10
	E-MODE	-4.836106E+02	-4.836106E+02	3.480934E-13
	SAMO-DE	-4.836106E+02	-4.836106E+02	4.144280E-06
	εDEag	-4.836106E+02	-4.836106E+02	3.890350E-13
	DE-DPS	-4.836106E+02	-4.836106E+02	1.258260E-10
	ECHT-ARMOR-DE	-4.836100E+02	-4.836100E+02	**0.000000E+00**
C06	CoBe-MmDE	-5.786624E+02	-5.786544E+02	3.676885E-02
	E-MODE	*-5.786624E+02*	**-5.786624E+02**	**3.248872E-13**
	SAMO-DE	*-5.786624E+02*	-5.786562E+02	9.352190E-03
	εDEag	-5.786581E+02	-5.786528E+02	3.627169E-03
	DE-DPS	*-5.786624E+02*	-5.786620E+02	8.053790E-04
	ECHT-ARMOR-DE	-5.786600E+02	-5.786600E+02	4.000000E-13
C07	CoBe-MmDE	*0.000000E+00*	5.338143E-28	2.615145E-27
	E-MODE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	SAMO-DE	*0.000000E+00*	7.762750E-23	3.880800E-22
	εDEag	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	DE-DPS	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	ECHT-ARMOR-DE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
C08	CoBe-MmDE	*0.000000E+00*	2.983917E+01	6.683484E+01
	E-MODE	*0.000000E+00*	1.006621E+01	3.029575E+00
	SAMO-DE	*0.000000E+00*	**2.520090E-25**	**1.260047E-24**
	εDEag	*0.000000E+00*	6.727528E+00	5.560648E+00
	DE-DPS	*0.000000E+00*	3.950387E+00	5.019040E+00
	ECHT-ARMOR-DE	*0.000000E+00*	7.526200E+00	5.000000E+00
C09	CoBe-MmDE	*0.000000E+00*	1.763261E-01	8.638180E-01
	E-MODE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	SAMO-DE	*0.000000E+00*	5.089752E+00	2.410828E+01
	εDEag	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	DE-DPS	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	ECHT-ARMOR-DE	*0.000000E+00*	1.763300E-01	8.800000E-01

TABLE 6.6: 10D statistical results comparison. Boldface remarks those best values and Italics remark those values that are equal than the other reported. Std denotes the standard deviation of 25 runs per test problem

Problem	Algorithm	Best	Mean	Std
C10	CoBe-MmDE	*0.000000E+00*	3.236223E-28	1.585419E-27
	E-MODE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	SAMO-DE	*0.000000E+00*	4.467656E-01	1.546298E+00
	εDEag	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	DE-DPS	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	ECHT-ARMOR-DE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
C11	CoBe-MmDE	**-1.522713E-03**	**-1.522713E-03**	6.903733E-15
	E-MODE	-1.522713E-03	-1.522713E-03	**8.902015E-17**
	SAMO-DE	-1.522710E-03	-1.522710E-03	3.667610E-09
	εDEag	-1.522713E-03	-1.522713E-03	6.341035E-11
	DE-DPS	-1.522710E-03	-1.522710E-03	3.142750E-12
	ECHT-ARMOR-DE	-1.522700E-03	-	4.400000E-02
C12	CoBe-MmDE	-4.265165E+02	-7.050982E+01	1.390978E+02
	E-MODE	-4.231332E+02	-9.461033E+01	1.476318E+02
	SAMO-DE	*-5.700899E+02*	-1.166134E+02	1.830005E+02
	εDEag	*-5.700899E+02*	**-3.367349E+02**	1.782166E+02
	DE-DPS	-1.992500E-01	-1.992500E-01	4.195990E-10
	ECHT-ARMOR-DE	-1.992500E-01	-1.992500E-01	**1.600000E-13**
C13	CoBe-MmDE	-6.842937E+01	-5.610207E+01	9.940010E+00
	E-MODE	*-6.842937E+01*	-6.842930E+01	2.965727E-04
	SAMO-DE	*-6.842937E+01*	*-6.842937E+01*	1.542877E-07
	εDEag	*-6.842937E+01*	-6.842936E+01	1.025960E-06
	DE-DPS	*-6.842937E+01*	*-6.842937E+01*	**0.000000E+00**
	ECHT-ARMOR-DE	-6.842900E+01	-6.716900E+01	2.100000E+00
C14	CoBe-MmDE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	E-MODE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	SAMO-DE	*0.000000E+00*	1.206380E-21	2.435928E-21
	εDEag	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	DE-DPS	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	ECHT-ARMOR-DE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
C15	CoBe-MmDE	*0.000000E+00*	7.195912E-01	1.648790E+00
	E-MODE	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**
	SAMO-DE	*0.000000E+00*	7.053800E-04	2.441385E-03
	εDEag	*0.000000E+00*	1.798978E-01	8.813156E-01
	DE-DPS	*0.000000E+00*	5.438912E-26	2.193290E-25
	ECHT-ARMOR-DE	*0.000000E+00*	2.824600E+00	1.600000E+00
C16	CoBe-MmDE	*0.000000E+00*	4.343286E-01	3.894433E-01
	E-MODE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	SAMO-DE	*0.000000E+00*	6.469600E-03	1.087042E-02
	εDEag	*0.000000E+00*	3.702054E-01	3.710479E-01
	DE-DPS	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	ECHT-ARMOR-DE	*0.000000E+00*	2.847800E-02	5.000000E-02
C17	CoBe-MmDE	3.133588E-02	1.089780E+00	1.135979E+00
	E-MODE	*0.000000E+00*	3.417493E-30	1.707206E-29
	SAMO-DE	*0.000000E+00*	1.265501E-23	3.217991E-23
	εDEag	1.463180E-17	1.249561E-01	1.937197E-01
	DE-DPS	*0.000000E+00*	1.061273E-24	3.887260E-24
	ECHT-ARMOR-DE	*0.000000E+00*	**3.697800E-33**	**3.100000E-33**
C18	CoBe-MmDE	*0.000000E+00*	1.185738E-22	5.782501E-22
	E-MODE	*0.000000E+00*	1.528418E-32	2.196087E-32
	SAMO-DE	1.173999E-23	4.447988E-19	6.636705E-19
	εDEag	3.731439E-20	9.678765E-19	1.811234E-18
	DE-DPS	*0.000000E+00*	3.209082E-23	7.124570E-23
	ECHT-ARMOR-DE	*0.000000E+00*	**0.000000E+00**	**0.000000E+00**

TABLE 6.7: 30D statistical results comparison. Boldface remarks those best values and Italics remark those values that are equal than the other reported. Std denotes the standard deviation of 25 runs per test problem

Problem	Algorithm	Best	Mean	Std
C01	CoBe-MmDE	-8.218626E-01	**-8.215353E-01**	1.047177E-03
	E-MODE	*-8.218840E-01*	-8.198980E-01	2.862970E-03
	SAMO-DE	-8.218838E-01	-8.143674E-01	4.766000E-03
	εDEag	-8.218255E-01	-8.208687E-01	**7.103893E-04**
	DE-DPS	*-8.218840E-01*	-8.212036E-01	1.796480E-03
	ECHT-ARMOR-DE	-8.180600E-01	-7.899200E-01	2.510000E-02
C02	CoBe-MmDE	-2.182167E+00	-2.090841E+00	6.780124E-02
	E-MODE	**-2.280970E+00**	**-2.276230E+00**	4.897230E-03
	SAMO-DE	-2.280962E+00	-2.276111E+00	**3.706000E-03**
	εDEag	-2.169248E+00	-2.151424E+00	1.197582E-02
	DE-DPS	-2.280671E+00	-2.244631E+00	5.205480E-02
	ECHT-ARMOR-DE	-2.260700E+00	-2.170600E+00	7.360000E-02
C03	CoBe-MmDE	2.060656E-06	1.555854E-05	1.363499E-05
	E-MODE	**0.000000E+00**	**3.123600E-25**	**5.729750E-25**
	SAMO-DE	4.599600E-24	4.826000E-22	1.146000E-21
	εDEag	2.867347E+01	2.883785E+01	8.047159E-01
	DE-DPS	1.620000E-19	1.847900E-13	4.179940E-13
	ECHT-ARMOR-DE	2.580100E-24	2.638000E+01	7.940000E+00
C04	CoBe-MmDE	4.323673E-04	2.667384E-02	1.271518E-01
	E-MODE	**-3.333330E-06**	**-3.333330E-06**	**2.459250E-16**
	SAMO-DE	-3.248000E-06	-2.411300E-06	4.492340E-07
	εDEag	4.698111E-03	8.162973E-03	3.067785E-03
	DE-DPS	-3.331800E-06	-3.312300E-06	2.039260E-08
	ECHT-ARMOR-DE	-3.332600E-06	8.371300E-02	2.890000E-01
C05	CoBe-MmDE	-4.637711E+02	-4.377643E+02	1.643445E+01
	E-MODE	**-4.836110E+02**	**-4.836110E+02**	**2.036340E-13**
	SAMO-DE	-4.836106E+02	-4.836106E+02	5.389910E-06
	εDEag	-4.531307E+02	-4.495460E+02	2.899105E+00
	DE-DPS	-4.836106E+02	-4.836106E+02	4.420740E-06
	ECHT-ARMOR-DE	-4.812200E+02	-4.333500E+02	1.460000E+02
C06	CoBe-MmDE	-5.257563E+02	-5.198528E+02	3.304011E+00
	E-MODE	**-5.306380E+02**	**-5.306380E+02**	**4.661020E-10**
	SAMO-DE	-5.306368E+02	-5.306155E+02	1.288050E-02
	εDEag	-5.285750E+02	-5.279068E+02	4.748378E-01
	DE-DPS	-5.306379E+02	-5.306329E+02	5.893640E-03
	ECHT-ARMOR-DE	-5.246500E+02	-4.893100E+02	1.320000E+02
C07	CoBe-MmDE	2.233625E-26	3.189299E-01	1.081544E+00
	E-MODE	*0.000000E+00*	**1.605130E-27**	**4.135470E-27**
	SAMO-DE	9.495250E-23	1.782790E-13	3.628040E-13
	εDEag	1.147112E-15	2.603632E-15	1.233430E-15
	DE-DPS	5.487860E-20	1.033590E-13	2.205400E-13
	ECHT-ARMOR-DE	*0.000000E+00*	1.078900E-25	2.200000E-25
C08	CoBe-MmDE	5.519847E-26	2.369536E+02	4.680700E+02
	E-MODE	*0.000000E+00*	**1.373980E-27**	**4.091250E-27**
	SAMO-DE	6.425810E-21	1.032930E-09	2.373300E-09
	εDEag	2.518693E-14	7.831464E-14	4.855177E-14
	DE-DPS	4.575719E-13	3.447568E-09	8.863660E-09
	ECHT-ARMOR-DE	*0.000000E+00*	2.010100E+01	4.700000E+01
C09	CoBe-MmDE	1.706569E-21	1.285319E+02	1.930639E+02
	E-MODE	*0.000000E+00*	**9.277620E-27**	**1.819230E-26**
	SAMO-DE	1.186170E-20	6.079667E+00	1.432599E+01
	εDEag	2.770665E-16	1.072140E+01	2.821923E+01
	DE-DPS	3.815800E-23	5.290590E-14	1.279390E-13
	ECHT-ARMOR-DE	*0.000000E+00*	4.611000E+00	2.310000E+01

TABLE 6.8: 30D statistical results comparison. Boldface remarks those best values and Italics remark those values that are equal than the other reported. Std denotes the standard deviation of 25 runs per test problem

Problem	Algorithm	Best	Mean	Std
C10	CoBe-MmDE	2.447880E-23	5.499459E+01	6.338358E+01
	E-MODE	**0.000000E+00**	**6.987910E-27**	**1.426400E-26**
	SAMO-DE	9.769000E-21	1.960780E+01	2.123749E+01
	εDEag	3.252002E+01	3.326175E+01	4.545577E-01
	DE-DPS	1.637780E-21	2.239280E-13	6.240580E-13
	ECHT-ARMOR-DE	6.020900E-13	6.553600E+01	1.070000E+02
C11	CoBe-MmDE	-3.923434E-04	-3.923223E-04	1.827407E-08
	E-MODE	*-3.923440E-04*	**-3.923440E-04**	**1.074800E-10**
	SAMO-DE	-3.923000E-04	-3.869000E-04	6.149660E-06
	εDEag	-3.268462E-04	-2.863882E-04	2.707605E-05
	DE-DPS	*-3.923440E-04*	-3.923423E-04	8.776880E-10
	ECHT-ARMOR-DE	-3.923400E-04	-	5.280000E-03
C12	CoBe-MmDE	-1.992635E-01	-8.785374E-02	4.174691E-01
	E-MODE	-1.992630E-01	**-1.992630E-01**	**2.902920E-09**
	SAMO-DE	-1.992598E-01	-1.992573E-01	1.321890E-06
	εDEag	-1.991453E-01	3.562330E+02	2.889253E+02
	DE-DPS	**-1.992635E-01**	-1.992600E-01	2.602870E-08
	ECHT-ARMOR-DE	-1.992600E-01	-1.607600E-01	1.930000E-01
C13	CoBe-MmDE	-6.842737E+01	-6.799795E+01	5.087294E-01
	E-MODE	-6.842860E+01	-6.516740E+01	1.918000E+00
	SAMO-DE	*-6.842940E+01*	**-6.819178E+01**	**3.891641E-01**
	εDEag	-6.642473E+01	-6.535310E+01	5.733005E-01
	DE-DPS	*-6.842940E+01*	-6.633141E+01	2.084930E+00
	ECHT-ARMOR-DE	-6.741600E+01	-6.464600E+01	1.970000E+00
C14	CoBe-MmDE	1.565253E-24	9.567897E-01	1.702620E+00
	E-MODE	**0.000000E+00**	**2.075870E-27**	**5.756600E-27**
	SAMO-DE	1.747000E-22	1.196910E-08	2.569420E-08
	εDEag	5.015863E-14	3.089407E-13	5.608409E-13
	DE-DPS	8.925796E-21	5.137406E-14	1.317910E-13
	ECHT-ARMOR-DE	1.580900E-27	6.613500E+02	2.470000E+03
C15	CoBe-MmDE	2.552281E-26	1.875528E+01	1.453617E+01
	E-MODE	**0.000000E+00**	**2.536820E-27**	**6.807380E-27**
	SAMO-DE	5.842400E-18	2.112810E+00	4.510670E+00
	εDEag	2.160345E+01	2.160376E+01	1.104834E-04
	DE-DPS	6.336258E-20	1.957805E-13	5.226870E-13
	ECHT-ARMOR-DE	1.171600E-04	3.131600E+08	1.200000E+09
C16	CoBe-MmDE	*0.000000E+00*	3.465391E-02	1.084541E-01
	E-MODE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	SAMO-DE	*0.000000E+00*	4.160700E-03	7.677900E-03
	εDEag	*0.000000E+00*	2.168404E-21	1.062297E-20
	DE-DPS	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
	ECHT-ARMOR-DE	*0.000000E+00*	*0.000000E+00*	*0.000000E+00*
C17	CoBe-MmDE	6.406286E-10	5.771224E-07	2.441370E-06
	E-MODE	1.954820E-32	**2.771470E-21**	**8.439140E-21**
	SAMO-DE	**0.000000E+00**	1.022630E-10	1.455150E-10
	εDEag	2.165719E-01	6.326487E+00	4.986691E+00
	DE-DPS	3.236013E-12	8.766191E-02	1.559660E-01
	ECHT-ARMOR-DE	3.356400E-16	4.033600E-01	3.510000E-01
C18	CoBe-MmDE	3.768347E-03	9.561410E-02	8.921330E-02
	E-MODE	1.277640E-28	1.340360E-20	6.550390E-20
	SAMO-DE	9.192800E-18	2.568100E-09	6.984800E-09
	εDEag	1.226054E+00	8.754569E+01	1.664753E+02
	DE-DPS	6.747743E-13	9.459914E-08	1.912240E-07
	ECHT-ARMOR-DE	**0.000000E+00**	**0.000000E+00**	**0.000000E+00**

TABLE 6.9: 95%-confidence Wilcoxon rank-sum test results for CoBe-MmDE and state-of-the-art algorithms. **Dim** means the search space dimension, while $w+$ and $w-$ mean the sum of positive and negative ranks, respectively. **Diff** denotes wheter there is a significant difference

Algorithm	**Dim**	**Criteria**	$w+$	$w-$	$Diff$
CoBe-MmDE – to – E-MODE	10	Best Fitness	17	11	$\approx$
		Average Fitness	50	28	$\approx$
	30	Best Fitness	167	4	$-$
		Average Fitness	171	0	$-$
CoBe-MmDE – to – SAMO-DE	10	Best Fitness	18	3	$\approx$
		Average Fitness	79	74	$\approx$
	30	Best Fitness	153	4	$-$
		Average Fitness	168	3	$-$
CoBe-MmDE – to – εDEag	10	Best Fitness	35	70	$\approx$
		Average Fitness	53	118	$\approx$
	30	Best Fitness	111	60	$\approx$
		Average Fitness	73	98	$\approx$
CoBe-MmDE – to – DE-DPS	10	Best Fitness	34	80	$\approx$
		Average Fitness	43	98	$\approx$
	30	Best Fitness	23	83	$\approx$
		Average Fitness	73	128	$\approx$
CoBe-MmDE – to – ECHT-ARMOR-DE	10	Best Fitness	32	11	$\approx$
		Average Fitness	18	77	$\approx$
	30	Best Fitness	78	87	$\approx$
		Average Fitness	107	19	$\approx$

Chapter 7

Conclusions and Future Work

During the development of this thesis, five studies on memetic algorithms based on differential evolution (MDE) were carried out. (1) Literature review, (2) study of local search performance influence, (3) study of local search depth influence, (4) analysis of Baldwin effect, and (5) the local search coordination proposed for a multimeme scheme.

In Chapter 4 several memetic algorithms based on differential evolution where commented. We can summarize those works in three parts. Most memetic approaches based on differential evolution made use of zero order local search operators, that is, free of derivatives. Likewise, the activation of local search operators was done through probabilistic mechanisms or mechanisms based on frequency. On the other hand, most of the MDE syntactic models maintain the use of local search throughout the evolutionary cycle of differential evolution, that is, the local search was not limited to only exploiting feasible regions of the search space.

Regarding multimeme schemes based on differential evolution, most used coordination mechanisms based on population diversity. Likewise, the probabilistic mechanisms for the activation of local search operators were a constant in this type of approach, where the focus was on adapting the local search intensity and also adaptively selecting candidate points to operate through the optimization process.

Finally, in real-world problems, the most used memetic approaches were those that use a single local search operator with probabilistic methods as activation mechanisms. In the same way, the use of parameters self-adaptation was frequent in those approaches.

From all of the above, it can be concluded that the versatility of differential evolution (DE) allows adapting multiple memetic schemes, of which, the computational effort is distributed either in the local search operators and DE as well.

Chapter 5 presented three empirical studies about the local search operators influence in Memetic DE Approaches for CNOPs. For the studies we designed three memetic algorithms based on different DE, starting from the same syntactic model, which consisted in the activation of a local search operator in a probabilistic manner just after applying the DE variation operators. Derived from the three studies, we can conclude the following points:

- The local search performance does not necessarily influence the final results of the memetic algorithm. Therefore, the usage of coordination mechanisms based only on the performance of the local search algorithm does not guarantee a competitive behavior of the memetic algorithm.

- Search depth in local search algorithms of zero order does not significantly affect the final results of the algorithm. Therefore, it is not necessary to carry out an excessive exploitation process during the application of the local search algorithm.
- The Lamarckian learning mechanism allows greater robustness in memetic algorithms based on DE for constrained search spaces.

Finally, Chapter 6 presented a local search coordination mechanism for a multimeme scheme based on DE, which was based on a cost-benefit approach (CoBe) to control the selection of a pool of local search operators. CoBe is able to modulate the local search activations robustly and efficiently, due to the stability of their behavior in different types of constrained problems. This is because the most significant benefit of local search operators in constrained problems is obtained at the beginning of the evolutionary process, and CoBe allows to gradually reduce the probability of activation.

Part of our future work includes the following:

- Test the proposed mechanism (CoBe) in multimeme approaches based on other evolutionary or swarm intelligence algorithms
- Test CoBe with a high number of local search operators.
- Test CoBe with other constraint handling techniques.

Appendix A

Problem Definitions CEC 2006

TABLE A.1: Features of the 24 test problems. **D** dimension of the problem; **LI** linear inequality constraint; **NI** nonlinear inequality constraint; **LE** linear equality constraint; **NE** nonlinear equality constraint. ρ estimates the ratio between the feasible region and the entire search space

Function	D	Type	ρ	LI	NI	LE	NE
g01	13	quadratic	0.0111%	9	0	0	0
g02	20	nonlinear	99.9971%	0	2	0	0
g03	10	polynomial	0.0000%	0	0	0	1
g04	5	quadratic	52.1230%	0	6	0	0
g05	4	cubic	0.0000%	2	0	0	3
g06	2	cubic	0.0066%	0	2	0	0
g07	10	quadratic	0.0003%	3	5	0	0
g08	2	nonlinear	0.8560%	0	2	0	0
g09	7	polynomial	0.5121%	0	4	0	0
g10	8	linear	0.0010%	3	3	0	0
g11	2	quadratic	0.0000%	0	0	0	1
g12	3	quadratic	4.7713%	0	1	0	0
g13	5	nonlinear	0.0000%	0	0	0	3
g14	10	nonlinear	0.0000%	0	0	3	0
g15	3	quadratic	0.0000%	0	0	1	1
g16	5	nonlinear	0.0204%	4	34	0	0
g17	6	nonlinear	0.0000%	0	0	0	4
g18	9	quadratic	0.0000%	0	13	0	0
g19	15	nonlinear	33.4761%	0	5	0	0
g20	24	linear	0.0000%	0	6	2	12
g21	7	linear	0.0000%	0	1	0	5
g22	22	linear	0.0000%	0	1	8	11
g23	9	linear	0.0000%	0	2	3	1
g24	2	linear	79.6556%	0	2	0	0

g01

Minimize:

$$f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to:

$$g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$
$$g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$$
$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$
$$g_4(\vec{x}) = -8x_1 + x_{10} \leq 0$$
$$g_5(\vec{x}) = -8x_2 + x_{11} \leq 0$$
$$g_6(\vec{x}) = -8x_3 + x_{12} \leq 0$$
$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leq 0$$
$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leq 0$$
$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

where the bounds are $0 \leq x_i \leq 1 (i = 1, \ldots, 9), 0 \leq x_i \leq 100 (i = 10, 11, 12)and0 \leq x13 \leq 1$

g02

Minimize:

$$f(\vec{x}) = -\left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

subject to:

$$g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$
$$g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

where $n = 20$ and $0 < x_i \leq 10 (i = 1, \ldots, n)$

g03

Minimize:

$$f(\vec{x}) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i$$

subject to:

$$h(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

where $n = 10$ and $0 \leq x_i \leq 1 (i = 1, \ldots, n)$.

g04

Minimize:

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

subject to:

$$g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$
$$g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$
$$g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$
$$g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$
$$g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$
$$g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

where $78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45 (i = 3, 4, 5)$

g05

Minimize:

$$f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

subject to:

$$g_1(\vec{x}) = -x_4 + x_3 - 0.55 \leq 0$$
$$g_2(\vec{x}) = -x_3 + x_4 - 0.55 \leq 0$$
$$h_3(\vec{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$
$$h_4(\vec{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$
$$h_5(\vec{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ and $-0.55 \leq x_4 \leq 0.55$.

g06

Minimize:

$$f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to:

$$g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$
$$g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$

g07

Minimize:

$$f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

subject to:

$$g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$
$$g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$
$$g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$
$$g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$
$$g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$
$$g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$
$$g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$
$$g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

where $-10 \leq x_i \leq 10 (i = 1, \ldots, 10)$

g08

Minimize:

$$f(\vec{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

subject to:

$$g_1(\vec{x}) = x_1^2 - x_2 + 1 \leq 0$$
$$g_2(\vec{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$

g09

Minimize:

$$f(\vec{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

subject to:

$$g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$
$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$
$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$
$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

where $-10 \leq x_i \leq 10 for (i = 1, \ldots, 7)$

g10

Minimize:

$$f(\vec{x}) = x_1 + x_2 + x_3$$

subject to:

$$\begin{aligned}
g_1(\vec{x}) &= -1 + 0.0025(x_4 + x_6) \leq 0 \\
g_2(\vec{x}) &= -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\
g_3(\vec{x}) &= -1 + 0.01(x_8 - x_5) \leq 0 \\
g_4(\vec{x}) &= -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0 \\
g_5(\vec{x}) &= -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0 \\
g_6(\vec{x}) &= -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0
\end{aligned}$$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000 (i = 2, 3)$ and $10 \leq x_i \leq 1000 (i = 4, \ldots, 8)$

g11

Minimize:

$$f(\vec{x}) = x_1^2 + (x_2 - 1)^2$$

subject to:

$$h(\vec{x}) = x_2 - x_1^2 = 0$$

where $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$

g12

Minimize:

$$f(\vec{x}) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$

subject to:

$$g(\vec{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

where $0 \leq x_i \leq 10 (i = 1, 2, 3)$ and $p, q, r = 1, 2, \ldots, 9$

g13

Minimize:

$$f(\vec{x}) = e^{x_1 x_2 x_3 x_4 x_5}$$

subject to:

$$\begin{aligned}
h_1(\vec{x}) &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0 \\
h_2(\vec{x}) &= x_2 x_3 - 5 x_4 x_5 = 0 \\
h_3(\vec{x}) &= x_1^3 + x_2^3 + 1 = 0
\end{aligned}$$

where $-2.3 \leq x_i \leq 2.3 (i = 1, 2)$ and $-3.2 \leq x_i \leq 3.2 (i = 3, 4, 5)$

g14

Minimize:

$$f(\vec{x}) = \sum_{i=1}^{10} x_i \left(c_i + ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$$

subject to:

$$\begin{aligned}
h_1(\vec{x}) &= x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0 \\
h_2(\vec{x}) &= x_4 + 2x_5 + x_6 + x_7 - 1 = 0 \\
h_3(\vec{x}) &= x_3 + x_7 + x_8 + 2x_9 + x_{10} = 0
\end{aligned}$$

where the bounds are $0 < x_i \leq 10(i = 1, \ldots, 10)$, and $c_1 = -6.089$, $c_2 = -17.164$, $c_3 = -34.054$, $c_4 = -5.914$, $c_5 = -24.721$, $c_6 = -14.986$, $c_7 = -24.1$, $c_8 = -10.708$, $c_9 = -26.662$, $c_{10} = -22.179$

g15

Minimize:

$$f(\vec{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

subject to:

$$h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$
$$h_2(\vec{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

where the bounds are $0 \leq x_i \leq 10(i = 1, 2, 3)$

g16

Minimize:

$$f(\vec{x}) = 0.000117y_{14} + 0.1365 + 0.0002358y_{13} + 0.000001502y_{16} + 0.0321y_{12} + 0.004324y_5 + 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$$

subject to:

$$
\begin{aligned}
g_1(\vec{x}) &= \frac{0.28}{0.72} y_5 - y_4 \leq 0 \\
g_2(\vec{x}) &= x_3 - 1.5 x_2 \leq 0 \\
g_3(\vec{x}) &= 3496 \frac{y_2}{c_{12}} - 21 \leq 0 \\
g_4(\vec{x}) &= 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0 \\
g_5(\vec{x}) &= 213.1 - y_1 \leq 0 \\
g_6(\vec{x}) &= y_1 - 405.23 \leq 0 \\
g_7(\vec{x}) &= 17.505 - y_2 \leq 0 \\
g_8(\vec{x}) &= y_2 - 1053.6667 \leq 0 \\
g_9(\vec{x}) &= 11.275 - y_3 \leq 0 \\
g_{10}(\vec{x}) &= y_3 - 35.03 \leq 0 \\
g_{11}(\vec{x}) &= 214.228 - y_4 \leq 0 \\
g_{12}(\vec{x}) &= y_4 - 665.585 \leq 0 \\
g_{13}(\vec{x}) &= 7.458 - y_5 \leq 0 \\
g_{14}(\vec{x}) &= y_5 - 584.463 \leq 0 \\
g_{15}(\vec{x}) &= 0.961 - y_6 \leq 0 \\
g_{16}(\vec{x}) &= y_6 - 265.916 \leq 0 \\
g_{17}(\vec{x}) &= 1.612 - y_7 \leq 0 \\
g_{18}(\vec{x}) &= y_7 - 7.046 \leq 0 \\
g_{19}(\vec{x}) &= 0.146 - y_8 \leq 0 \\
g_{20}(\vec{x}) &= y_8 - 0.222 \leq 0 \\
g_{21}(\vec{x}) &= 107.99 - y_9 \leq 0 \\
g_{22}(\vec{x}) &= y_9 - 273.366 \leq 0 \\
g_{23}(\vec{x}) &= 922.693 - y_{10} \leq 0 \\
g_{24}(\vec{x}) &= y_{10} - 1286.105 \leq 0 \\
g_{25}(\vec{x}) &= 926.832 - y_{11} \leq 0 \\
g_{26}(\vec{x}) &= y_{11} - 1444.046 \leq 0 \\
g_{27}(\vec{x}) &= 18.766 - y_{12} \leq 0 \\
g_{28}(\vec{x}) &= y_{12} - 537.141 \leq 0 \\
g_{29}(\vec{x}) &= 1072.163 - y_{13} \leq 0 \\
g_{30}(\vec{x}) &= y_{13} - 3247.039 \leq 0 \\
g_{31}(\vec{x}) &= 8961.448 - y_{14} \leq 0 \\
g_{32}(\vec{x}) &= y_{14} - 26844.086 \leq 0 \\
g_{33}(\vec{x}) &= 0.063 - y_{15} \leq 0 \\
g_{34}(\vec{x}) &= y_{15} - 0.386 \leq 0 \\
g_{35}(\vec{x}) &= 71084.33 - y_{16} \leq 0 \\
g_{36}(\vec{x}) &= -140000 + y_{16} \leq 0 \\
g_{37}(\vec{x}) &= 2802713 - y_{17} \leq 0 \\
g_{38}(\vec{x}) &= y_{17} - 12146108 \leq 0
\end{aligned}
$$

where:

$$
\begin{aligned}
y_1 &= x_2 + x_3 + 41.6 \\
c_1 &= 0.024x_4 - 4.62 \\
y_2 &= \frac{12.5}{c_1} + 12 \\
c_2 &= 0.0003535x_1^2 + 0.5311x_1 + 0.08705y_2x_1 \\
c_3 &= 0.052x_1 + 78 + 0.002377y_2x_1 \\
y_3 &= \frac{c_2}{c_3} \\
y_4 &= 19y_3 \\
c_4 &= 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3 \\
c_5 &= 100x_2 \\
c_6 &= x_1 - y_3 - y_4 \\
c_7 &= 0.950 - \frac{c_4}{c_5} \\
y_5 &= c_6c_7 \\
y_6 &= x_1 - y_5 - y_4 - y_3 \\
c_8 &= (y_5 + y_4)0.995 \\
y_7 &= \frac{c_8}{y_1} \\
y_8 &= \frac{c_8}{3798} \\
c_9 &= y_7 - \frac{0.0663y_7}{y_8} - 0.3153 \\
y_9 &= \frac{96.82}{c_9} + 0.321y_1 \\
y_{10} &= 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6 \\
y_{11} &= 1.71x_1 - 0.452y_4 + 0.580y_3 \\
c_{10} &= \frac{12.3}{752.3} \\
c_{11} &= (1.75y_2)(0.995x_1) \\
c_{12} &= 0.995y_{10} + 1998 \\
y_{12} &= c_{10}x_1 + \frac{c_{11}}{c_{12}} \\
y_{13} &= c_{12} - 1.75y_2 \\
y_{14} &= 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5} \\
c_{13} &= 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095 \\
y_{15} &= \frac{y_{13}}{c_{13}} \\
y_{16} &= 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13} \\
c_{14} &= 2324y_{10} - 28740000y_2 \\
y_{17} &= 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}} \\
c_{15} &= \frac{y_{13}}{y_{15}} - \frac{y_13}{0.52} \\
c_{16} &= 1.104 - 0.72y_{15} \\
c_{17} &= y_9 + x_5
\end{aligned}
$$

and where the bounds are $704.4148 \leq x_1 \leq 906.3855$, $68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75$, $193 \leq x_4 \leq 287.0966$ and $25 \leq x_5 \leq 84.1988$.

g17

Minimize:

$$f(\vec{x}) = f(x_1) + f(x_2)$$

where:

$$f_1(x_1) = \begin{cases} 30x_1 & 0 \leq x_1 < 300 \\ 31x_1 & 300 \leq x_1 < 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \leq x_2 < 100 \\ 29x_2 & 100 \leq x_2 < 200 \\ 30x_2 & 200 \leq x_2 < 1000 \end{cases}$$

subject to:

$$h_1(\vec{x}) = -x_1 + 300 - \frac{x_3 x_4}{131.078} cos(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078} cos(1.47588)$$
$$h_2(\vec{x}) = -x_2 - \frac{x_3 x_4}{131.078} cos(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} cos(1.47588)$$
$$h_3(\vec{x}) = -x_5 - \frac{x_3 x_4}{131.078} sin(1.48477 + x_6) + \frac{0.90798 x_4^2}{131.078} sin(1.47588)$$
$$h_4(\vec{x}) = 200 - \frac{x_3 x_4}{131.078} sin(1.48477 - x_6) + \frac{0.90798 x_3^2}{131.078} sin(1.47588)$$

where the bounds are $0 \leq x_1 \leq 400$, $0 \leq x_2 \leq 1000$, $340 \leq x_3 \leq 420$, $340 \leq x_4 \leq 420$, $-1000 \leq x_5 \leq 1000$ and $0 \leq x_6 \leq 0.5236$

g18

Minimize:

$$f(\vec{x}) = -0.5(x_1 x_4 - x_2 x_3 + x_3 x_9 - x_5 x_9 + x_5 x_8 - x_6 x_7)$$

subject to:

$$g_1(\vec{x}) = x_3^2 + x_4^2 - 1 \leq 0$$
$$g_2(\vec{x}) = x_9^2 - 1 \leq 0$$
$$g_3(\vec{x}) = x_5^2 + x_6^2 - 1 \leq 0$$
$$g_4(\vec{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \leq 0$$
$$g_5(\vec{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$$
$$g_6(\vec{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$$
$$g_7(\vec{x}) = (x_3 - x_5)^2 + (x_2 - x_6)^2 - 1 \leq 0$$
$$g_8(\vec{x}) = (x_3 - x_7)^2 + (x_2 - x_8)^2 - 1 \leq 0$$
$$g_9(\vec{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \leq 0$$
$$g_{10}(\vec{x}) = x_2 x_3 - x_1 x_4 \leq 0$$
$$g_{11}(\vec{x}) = -x_3 x_9 \leq 0$$
$$g_{12}(\vec{x}) = x_5 x_9 \leq 0$$
$$g_{13}(\vec{x}) = x_6 x_7 - x_5 x_8 \leq 0$$

where the bounds are $-10 \leq x_i \leq 10 (i = 1, \ldots, 8)$ and $0 \leq x_9 \leq 20$

g19

Minimize:

$$f(\vec{x}) = \sum_{j=1}^{5} \sum_{i=1}^{5} c_{i,j} x_{(10+j)} + 2 \sum_{j=1}^{5} d_j x_{(10+j)}^3 - \sum_{i=1}^{10} b_i x_i$$

subject to:

$$g(\vec{x}) = -2 \sum_{i=1}^{5} c_{i,j} x_{(10+j)} - e_j + \sum_{i=1}^{10} a_{i,j} x_i \leq 0 \quad j = 1, \ldots, 5$$

where $b = [-40, -2, -0.25, -4, -4, -1, -40, -60, 5, 1]$. The bounds are $0 \leq x_i \leq 10 (i = 1, \ldots, 15)$

g20

Minimize:

$$f(\vec{x}) = \sum_{i=1}^{24} a_i x_i$$

subject to:

$$\begin{aligned}
g_i(\vec{x}) &= \frac{(x_i + x_{(i+12)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 1, 2, 3 \\
g_i(\vec{x}) &= \frac{(x_{(i+3)} + x_{(i+15)})}{\sum_{j=1}^{24} x_j + e_i} \leq 0 \quad i = 4, 5, 6 \\
h_i(\vec{x}) &= \frac{x_{(i+12)}}{b_{(i+12)} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0 \quad i = 1, \ldots, 12 \\
h_{13}(\vec{x}) &= \sum_{i=1}^{24} x_i - 1 = 0 \\
h_{14}(\vec{x}) &= \sum_{i=1}^{12} \frac{x_i}{d_i} + k \sum_{i=13}^{24} \frac{x_i}{b_i} - 1.671 = 0
\end{aligned}$$

where $k = (0.7302)(530)(\frac{14.7}{40})$. The bounds are $0 \leq x_i \leq 10 (i = 1, \ldots, 24)$

g21

Minimize:

$$f(\vec{x}) = x_1$$

subject to:

$$\begin{aligned}
g_1(\vec{x}) &= -x_1 + 35 x_2^{0.62} + 35 x_3^{0.6} \leq 0 \\
h_1(\vec{x}) &= -300 x_3 + 7500 x_5 - 7500 x_6 - 25 x_4 x_5 + 25 x_4 x_6 + x_3 x_4 = 0 \\
h_2(\vec{x}) &= 100 x_2 + 155.365 x_4 + 2500 x_7 - x_2 x_4 - 25 x_4 x_7 - 15536.5 = 0 \\
h_3(\vec{x}) &= -x_5 + ln(-x_4 + 900) = 0 \\
h_4(\vec{x}) &= -x_6 + ln(x_4 + 300) = 0 \\
h_5(\vec{x}) &= -x_7 + ln(-2 x_4 + 700) = 0
\end{aligned}$$

where the bounds are $0 \leq x_1 \leq 1000$, $0 \leq x_2, x_3 \leq 40$, $100 \leq x_4 \leq 300$, $6.3 \leq x_5 \leq 6.7$, $5.9 \leq x_6 \leq 6.4$ and $4.5 \leq x_7 \leq 6.25$

g22

Minimize:

$$f(\vec{x}) = x_1$$

subject to:

$$\begin{aligned}
g_1(\vec{x}) &= -x_1 + x_2^{0.62} + x_3^{0.6} + x_4^{0.6} \leq 0 \\
h_1(\vec{x}) &= x_5 - 100000x_8 + 1 \times 10^7 = 0 \\
h_2(\vec{x}) &= x_6 + 100000x_8 - 100000x9 = 0 \\
h_3(\vec{x}) &= x_7 + 100000x_9 - 5 \times 10^7 = 0 \\
h_4(\vec{x}) &= x_5 + 100000x_{10} - 3.3 \times 10^7 = 0 \\
h_5(\vec{x}) &= x_6 + 100000x_{11} - 4.4 \times 10^7 = 0 \\
h_6(\vec{x}) &= x_7 + 100000x_{12} - 6.6 \times 10^7 = 0 \\
h_7(\vec{x}) &= x_5 - 120x_2x_{13} = 0 \\
h_8(\vec{x}) &= x_6 - 80x_3x_{14} = 0 \\
h_9(\vec{x}) &= x_7 - 40x_4x_{15} = 0 \\
h_{10}(\vec{x}) &= x_8 - x_{11} + x_{16} = 0 \\
h_{11}(\vec{x}) &= x_9 - x_{12} + x_{17} = 0 \\
h_{12}(\vec{x}) &= -x_{18} + ln(x_{10} - 100) = 0 \\
h_{13}(\vec{x}) &= -x_{19} + ln(-x_8 + 300) = 0 \\
h_{14}(\vec{x}) &= -x_{20} + ln(x_{16}) = 0 \\
h_{15}(\vec{x}) &= -x_{21} + ln(-x_9 + 400) = 0 \\
h_{16}(\vec{x}) &= -x_{22} + ln(x_{17}) = 0 \\
h_{17}(\vec{x}) &= -x_8 - x_{10} + x_{13}x_{18} - x_{13}x_{19} + 400 = 0 \\
h_{18}(\vec{x}) &= x_8 - x_9 - x_{11} + x_{14}x_{20} - x_{14}x_{21} + 400 = 0 \\
h_{19}(\vec{x}) &= x_9 - x_{12} - 4.60517x_{15} + x_{15}x_{22} + 100 = 0
\end{aligned}$$

where the bounds are $0 \leq x_1 \leq 20000$, $0 \leq x_2, x_3, x_4 \leq 1 \times 10^6$, $0 \leq x_5, x_6, x_7 \leq 4 \times 10^7$, $100 \leq x_8 \leq 299.99$, $100 \leq x_9 \leq 399.99$, $100.01 \leq x_{10} \leq 300$, $100 \leq x_{11} \leq 400$, $100 \leq x_{12} \leq 600$, $0 \leq x_{13}, x_{14}, x_{15} \leq 500$, $0.01 \leq x_{16} \leq 300$, $0.01 \leq x_{17} \leq 400$, $-4.7 \leq x_{18}, x_{19}, x_{20}, x_{21}, x_{22} \leq 6.25$

g23

Minimize:

$$f(\vec{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$$

subject to:

$$\begin{aligned}
g_1(\vec{x}) &= x_9x_3 + 0.02x_6 - 0.025x_5 \leq 0 \\
g_2(\vec{x}) &= x_9x_4 + 0.02x_7 - 0.015x_8 \leq 0 \\
h_1(\vec{x}) &= x_1 + x_2 - x_3 - x_4 = 0 \\
h_2(\vec{x}) &= 0.03x_1 + 0.01x_2 - x_9(x_3 + x_4) = 0 \\
h_3(\vec{x}) &= x_3 + x_6 - x_5 = 0 \\
h_4(\vec{x}) &= x_4 + x_7 - x_8 = 0
\end{aligned}$$

where the bounds are $0 \leq x_1, x_2, x_6 \leq 300$, $0 \leq x_3, x_5, x_7 \leq 100$, $0 \leq x_4, x_8 \leq 200$ and $0.01 \leq x_9 \leq 0.03$

g24

Minimize:

$$f(\vec{x}) = -x_1 - x_2$$

subject to:

$$g_1(\vec{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0$$
$$g_2(\vec{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0$$

where the bounds $are 0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 4$

Appendix B

Problem Definitions CEC 2010

TABLE B.1: Details of 18 test problems. D is the number of decision variables. S, NS and R are Separable, Non Separable and Rotated functions, respectively.

Problem	Range	Type of Objective	Number of Constraints	
			Equality	Inequality
C01	$[0, 10]^D$	Non Separable	-	2-NS
C02	$[-5.12, 5.12]^D$	Separable	1-S	2-S
C03	$[-1000, 1000]^D$	Non Separable	1-NS	-
C04	$[-50, 50]^D$	Separable	2-S / 2-NS	-
C05	$[-600, 600]^D$	Separable	2-S	-
C06	$[-600, 600]^D$	Separable	2-R	-
C07	$[-140, 140]^D$	Non Separable	-	1-S
C08	$[-140, 140]^D$	Non Separable	-	1-R
C09	$[-500, 500]^D$	Non Separable	1-S	-
C10	$[-500, 500]^D$	Non Separable	1-R	-
C11	$[-100, 100]^D$	Rotated	1-NS	-
C12	$[-1000, 1000]^D$	Separable	1-NS	1-S
C13	$[-500, 500]^D$	Separable	-	2-S / 1-NS
C14	$[-1000, 1000]^D$	Non Separable	-	3-S
C15	$[-1000, 1000]^D$	Non Separable	-	3-R
C16	$[-10, 10]^D$	Non Separable	2-S	1-S / 1-NS
C17	$[-10, 10]^D$	Non Separable	1-S	2-NS
C18	$[-50, 50]^D$	Non Separable	1-S	1-S

C01

Minimize:

$$f(x) = -\left|\sum_{i=1}^{D} cos^4(z_i) - 2\prod_{i=1}^{D} cos^2(z_i)\right|$$

$$z = x - o$$

subject to:

$$g_1(x) = 0.75 - \prod_{i=1}^{D} z_i \leq 0$$
$$g_2(x) = \sum_{i=1}^{D} z_i - 7.5D \leq 0$$
$$x \in [0, 10]^D$$

C02

Minimize:

$$f(x) = max(z)$$
$$z = x - o$$
$$y = z - 0.5$$

subject to:

$$g_1(x) = 10 - \frac{1}{D}\sum_{i=1}^{D}\left[z_i^2 - 10cos(2\Pi z_i) + 10\right] \leq 0$$
$$g_2(x) = \frac{1}{D}\sum_{i=1}^{D}\left[z_i^2 - 10cos(2\Pi z_i) + 10\right] - 15 \leq 0$$
$$h(x) = \frac{1}{D}\sum_{i=1}^{D}\left[y_i^2 - 10cos(2\Pi y_i) + 10\right] - 20 = 0$$
$$x \in [-5.12, 5.12]^D$$

C03

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x - o$$

subject to:

$$h(x) = \sum_{i=1}^{D-1}(z_i - z_{i+1})^2 = 0$$
$$x \in [-1000, 1000]^D$$

C04

Minimize:

$$f(x) = max(z)$$
$$z = x - o$$

subject to:

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(z_i cos(\sqrt{|z_i|})) = 0$$
$$h_2(x) = \sum_{i=1}^{D/2-1}(z_i - z_{i+1})^2 = 0$$
$$h_3(x) = \sum_{i=1}^{D-1}(z_i^2 - z_{i+1})^2 = 0$$
$$h_4(x) = \sum_{i=1}^{D} z = 0$$
$$x \in [-50, 50]^D$$

C05

Minimize:

$$f(x) = max(z)$$
$$z = x - o$$

subject to:

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i sin(\sqrt{|z_i|})) = 0$$
$$h_2(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i cos(0.5\sqrt{|z_i|})) = 0$$
$$x \in [-600, 600]^D$$

C06

Minimize:

$$f(x) = max(z)$$
$$z = x - o$$
$$y = (x + 483.6106156535 - o)M - 483.6106156535$$

subject to:

$$h_1(x) = \frac{1}{D}\sum_{i=1}^{D}(-y_i sin(\sqrt{|y_i|})) = 0$$
$$h_2(x) = \frac{1}{D}\sum_{i=1}^{D}(-y_i cos(0.5\sqrt{|y_i|})) = 0$$
$$x \in [-600, 600]^D$$

C07

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = x - o$$

subject to:

$$g(x) = 0.5 - \exp\left(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(0.1y)\right) + \exp(1) \leq 0$$
$$x \in [-140, 140]^D$$

C08

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = (x - o)M$$

subject to:

$$g(x) = 0.5 - \exp\left(-0.1\sqrt{\frac{1}{D}\sum_{i=1}^{D} y_i^2}\right) - 3\exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(0.1y)\right) + \exp(1) \leq 0$$
$$x \in [-140, 140]^D$$

C09

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = x - o$$

subject to:

$$h(x) = \sum_{i=1}^{D}(ysin(\sqrt{|y_i|})) = 0$$
$$x \in [-500, 500]^D$$

C10

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2)$$
$$z = x + 1 - o$$
$$y = (x - o)M$$

subject to:

$$h(x) = \sum_{i=1}^{D}(ysin(\sqrt{|y_i|})) = 0$$
$$x \in [-500, 500]^D$$

C11

Minimize:

$$f(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i cos(2\sqrt{|z_i|}))$$
$$z = (x - o)M$$
$$y = x + 1 - o$$

subject to:

$$h(x) = \sum_{i=1}^{D-1}(100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2) = 0$$
$$x \in [-100, 100]^D$$

C12

Minimize:

$$f(x) = \frac{1}{D}\sum_{i=1}^{D}(z_i sin(\sqrt{|z_i|}))$$
$$z = x - o$$

subject to:

$$h(x) = \sum i = 1^{D-1}(z_i^2 - z_{i+1})^2 = 0$$
$$g(x) = \sum i = 1^{D}(z - 100cos(0.1z) + 10) \leq 0$$
$$x \in [-1000, 1000]^D$$

C13

Minimize:

$$f(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i sin(\sqrt{|z_i|}))$$
$$z = x - o$$

subject to:

$$g_1(x) = -50 + \frac{1}{100D}\sum_{i}^{D} z_i^2 \leq 0$$
$$g_2(x) = \frac{50}{D}\sum_{i=1}^{D} sin\left(\frac{1}{50}\Pi z\right) \leq 0$$
$$g_3(x) = 75 - 50\left(\sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1\right) \leq 0$$
$$x \in [-500, 500]^D$$

C14

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z-1)^2)$$
$$z = x + 1 - o$$
$$y = x - o$$

subject to:

$$g_1(x) = \sum_{i=1}^{D}(-y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_2(x) = \sum_{i=1}^{D}(y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_3(x) = \sum_{i=1}^{D}(y_i sin(\sqrt{y_i})) - 10D \leq 0$$
$$x \in [-1000, 1000]^D$$

C15

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z-1)^2)$$
$$z = x + 1 - o$$
$$y = (x - o)M$$

subject to:

$$g_1(x) = \sum_{i=1}^{D}(-y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_2(x) = \sum_{i=1}^{D}(y_i cos(\sqrt{y_i})) - D \leq 0$$
$$g_3(x) = \sum_{i=1}^{D}(y_i sin(\sqrt{y_i})) - 10D \leq 0$$
$$x \in [-1000, 1000]^D$$

C16

Minimize:

$$f(x) = \sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$$
$$z = x - o$$

subject to:

$$g_1(x) = \sum_{i=1}^{D}[z_i^2 - 100cos(\Pi z_i) + 10] \leq 0$$
$$g_2(x) = \prod_{i=1}^{D} z_i \leq 0$$
$$h_1(x) = \sum_{i=1}^{D}(z_i sin(\sqrt{|z_i|})) = 0$$
$$h_2(x) = \sum_{i=1}^{D}(-z_i sin(\sqrt{|z_i|})) = 0$$
$$x \in [-10, 10]^D$$

C17

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(z_i - z_{i+1})^2$$
$$z = x - o$$

subject to:

$$g_1(x) = \prod_{i=1}^{D} z_i \leq 0$$
$$g_2(x) = \sum_{i=1}^{D} z_i \leq 0$$
$$h_1(x) = \sum_{i=1}^{D}(z_i sin(4\sqrt{z_i})) = 0$$
$$x \in [-10, 10]^D$$

C18

Minimize:

$$f(x) = \sum_{i=1}^{D-1}(z_i - z_{i+1})^2$$
$$z = x - o$$

subject to:

$$g(x) = \frac{1}{D}\sum_{i=1}^{D}(-z_i sin(\sqrt{z_i})) \leq 0$$
$$h(x) = \frac{1}{D}\sum_{i=1}^{D}(z_i sin(\sqrt{z_i})) = 0$$
$$x \in [-50, 50]^D$$

Bibliography

[1] K. Deb, *Optimization for Engineering Design Algorithms and Examples*. Prentice-Hall of India, fisrt edition ed., 1995.

[2] D. Himmelblau, *Applied Nonlinear Programming*. Mcgraw-Hill, 1972.

[3] E. Mezura-Montes, O. Cetina-Domínguez, and B. Hernández-Ocaña, *Mecatrónica*, ch. Nuevas Heurísticas Inspiradas en la Naturaleza para Optimización Numérica, pp. 249–272. IPN, 2010.

[4] G. EPPEN, Á. Ruiz, and G. García, *Investigación de Operaciones en la Ciencia Administrativa: Construcción de Modelos para la Toma de Decisiones con Hojas de Cálculo Electrónicas*, ch. Optimización no Lineal, pp. 328–378. Pearson educación, Prentice Hall Hispanoamericana, S.A., 2000.

[5] M. Cánovas, V. Navarro, and M. Orts, *Optimización matemática aplicada.: Enunciados, ejercicios y aplicaciones del mundo real con matlab*. Editorial Club Universitario, 2011.

[6] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, pp. 341–359, 1997.

[7] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution – An updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[8] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[9] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech Concurrent Computation Program, C3P Report*, vol. 826, p. 1989, 1989.

[10] F. Neri and V. Tirronen, *Recent advances in differential evolution: a survey and experimental analysis*, vol. 33. 2010.

[11] X. Li and M. Yin, "Parameter estimation for chaotic systems by hybrid differential evolution algorithm and artificial bee colony algorithm," *Nonlinear Dynamics*, vol. 77, no. 1-2, pp. 61–71, 2014.

[12] N. Krasnogor, *Studies on the theory and design space of memetic algorithms.* PhD thesis, University of the West of England at Bristol, 2002.

[13] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, pp. 474–488, 2005.

[14] T. Takahama and S. Sakai, "Constrained Optimization by the ϵ Constrained Differential Evolution with an Archive and Gradient-Based Mutation," No. 1, (Barcelona, Spain), pp. 18–23, 2010.

[15] T. Takahama and S. Sakai, "Constrained Optimization by the Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites," *IEEE Congress on Evolution Computation*, pp. 1–8, 2006.

[16] S. Hernandez, G. Leguizamón, and E. Mezura-Montes, "A hybrid version of differential evolution with two differential mutation operators applied by stages," *2013 IEEE Congress on Evolutionary Computation*, pp. 2895–2901 „ 2013.

[17] S. Dominguez-Isidro, E. Mezura-Montes, and G. Leguizamón, "Memetic Differential Evolution for Constrained Numerical Optimization Problems," *2013 IEEE Congress on Evolutionary Computation*, pp. 2996–3003, 2013.

[18] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Self-adaptive differential evolution incorporating a heuristic mixing of operators," *Computational Optimization and Applications*, vol. 54, no. 3, pp. 771–790, 2013.

[19] S. Dominguez-Isidro, E. Mezura-Montes, and G. Leguizamón, "Performance Comparison of Local Search Operators in Differential Evolution for Constrained Numerical Optimization Problems," in *Differential Evolution (SDE), 2014 IEEE Symposium on*, (Orlado FL), pp. 1–8, IEEE, 2014.

[20] A. Menchaca-Mendez and C. A. Coello Coello, "A New Proposal to Hybridize the Nelder-Mead Method to a Differential Evolution Algorithm for Constrained Optimization," *2009 IEEE Congress on Evolutionary Computation*, pp. 2598–2605, 2009.

[21] M. Pescador-Rojas and C. A. C. Coello, "A memetic algorithm with simplex crossover for solving constrained optimization problems," in *World Automation Congress 2012*, pp. 1–6, June 2012.

[22] W. Yi, Y. Zhou, L. Gao, X. Li, and C. Zhang, "Engineering design optimization using an improved local search based epsilon differential evolution algorithm," *Journal of Intelligent Manufacturing*, p. ., 2016.

[23] A. Maesani, G. Iacca, and D. Floreano, "Memetic Viability Evolution for Constrained Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, pp. 125–144, 2015.

[24] N. Krasnogor and J. Smith, "Emergence of Profitable Search Strategies Based on a Simple Inheritance Mechanism," in *Genetic and Evolutionary Computation Conference (GECCO-2001)*, pp. 432–439, 2001.

[25] D. Sudholt, "Local Search in Memetic Algorithms: the Impact of the Local Search Frequency," *Algorithms and Computation*, pp. 359–368, 2006.

[26] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi, "An enhanced memetic differential evolution in filter design for defect detection in paper production," *Evolutionary Computation*, vol. 16, pp. 529–555, dec 2008.

[27] F. Neri and V. Tirronen, "On memetic Differential Evolution frameworks: A study of advantages and limitations in hybridization," *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 2135–2142, 2008.

[28] A. Caponio, F. Neri, and V. Tirronen, "Super-fit control adaptation in memetic differential evolution frameworks," *Soft Computing*, vol. 13, no. 8-9, pp. 811–831, 2009.

[29] G. Iacca, F. Neri, F. Caraffini, and P. N. Suganthan, *A Differential Evolution Framework with Ensemble of Parameters and Strategies and Pool of Local Search Algorithms*, pp. 615–626. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[30] N. R. Sabar, J. Abawajy, and J. Yearwood, "Heterogeneous cooperative coevolution memetic differential evolution algorithm for big data optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, pp. 315–327, April 2017.

[31] A. S. S. M. Barkat Ullah and R. Sarker, "An Agent-based Memetic Algorithm (AMA) for Solving Constrained Optimization Problems," *Evolutionary Computation*, pp. 999–1006, 2007.

[32] A. S. S. M. Barkat Ullah, R. Sarker, D. Cornforth, and C. Lokan, "AMA: a new approach for solving constrained real-valued optimization problems," *Soft Computing*, vol. 13, no. 8-9, pp. 741–762, 2009.

[33] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 561–579, Aug 2010.

[34] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173 – 194, 2011.

[35] T. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 4, pp. 284 – 294, sep 2000.

[36] H. A. Taha, *Investigación de Operaciones*. Pearson, novena ed., 2012.

[37] R. Hooke and T. A. Jeeves, ""direct search" solution of numerical and statistical problems," *J. ACM*, vol. 8, pp. 212–229, apr 1961.

[38] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[39] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 3 ed., 2007.

[40] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 ed., 2003. Spanish.

[41] C. Darwin, *The Origin of Species by Means of Natural Selection or the preservation of Favored Races in the Struggle for life*. New York: Random House, 1993. (Publicado originalmente en 1929).

[42] A. P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. Wiley, 2005.

[43] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.

[44] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. New York: Oxford University Press, 1996.

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

[46] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[47] R. Poli, W. B. Langdon, and N. F. McPhee, *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.

[48] H.-P. P. Schwefel, *Evolution and Optimum Seeking: The Sixth Generation*. New York, NY, USA: John Wiley & Sons, Inc., 1993.

[49] H.-G. Beyer, *The theory of Evolution Strategies*. Berlin: Springer, 2001.

[50] L. J. Fogel, *Intelligence Through Simulated Evolution. Forty years of Evolutionary Programming*. New York: John Wiley & Sons, 1999.

[51] R. Tanabe and A. Fukunaga, *Reevaluating Exponential Crossover in Differential Evolution*, pp. 201–210. Cham: Springer International Publishing, 2014.

[52] J. Kennedy and R. C. Eberhart, *Swarm intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[53] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Transactions of Systems, Man and Cybernetics-Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[54] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[55] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems*, vol. 22, pp. 52–67, Jun 2002.

[56] E. Mezura-Montes and B. Hernández-Ocaña, "Modified bacterial foraging optimization for engineering design," in *Proceedings of the Artificial Neural Networks in Enginnering Conference (ANNIE'2009)* (in Cihan H. Dagli et al. (editors), ed.), vol. 19 of *ASME Press Series*, pp. 357–364, Intelligent Engineering Systems Through Artificial Neural Networks, Noviembre 2009.

[57] N. Cruz-Cortés, "Handling constraints in global optimization using artificial immune systems: A survey," in *Constraint-Handling in Evolutionary Optimization* (E. Mezura-Montes, ed.), vol. 198 of *Studies in Computational Intelligence*, pp. 237–262, Springer Berlin / Heidelberg.

[58] D. Dasgupta, "Advances in artificial immune systems," *IEEE Computational Intelligence Magazine*, vol. 1, pp. 40–49, Nov 2006.

[59] N. Cruz-Cortés and C. A. Coello-Coello, "Un sistema inmune artificial para solucionar problemas de optimización multiobjetivo," tech. rep., Departamento de Ingeniería Eléctrica Sección de Computación, Instituto Politécnico Nacional, México DF., 2003. In Spanish.

[60] E. Mezura-Montes, *Constraint-Handling in Evolutionary Optimization*. Springer Publishing Company, Incorporated, 1st ed., 2009.

[61] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2-4, pp. 311–338, 2000.

[62] T. Takahama, S. Sakai, and N. Iwane, "Constrained optimization by the ε-constrained hybrid algorithm of particle swarm optimization and genetic algorithm," in *AI 2005: Advances in Artificial Intelligence* (S. Zhang and R. Jarvis, eds.), vol. 3809 of *Lecture Notes in Computer Science*, pp. 389–400, Springer Berlin Heidelberg, 2005.

[63] C. Houck, J. Joines, and M. Kay, "Utilizing Lamarckian evolution and the Baldwin effect in hybrid genetic algorithms," *NCSU-IE Technical Report*, pp. 96–01, 1996.

[64] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[65] R. Dawkins, *The Selfish Gene*. New York: Oxford University Press, 1976.

[66] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, 2004.

[67] N. Krasnogor, B. P. Blackburne, E. K. Burke, and J. D. Hirst, "Multimeme algorithms for protein structure prediction," in *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, PPSN VII, (London, UK, UK), pp. 769–778, Springer-Verlag, 2002.

[68] S. Muelas, A. La Torre, and J.-M. Peña, "A memetic differential evolution algorithm for continuous optimization," in *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, ISDA '09, (Washington, DC, USA), pp. 1080–1084, IEEE Computer Society, 2009.

[69] A. Mandal, A. Das, P. Mukherjee, S. Das, and P. Suganthan, "Modified differential evolution with local search algorithm for real world optimization," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 1565 –1572, june 2011.

[70] F. Solis and R. Wets, "Minimization by random search techniques," *Math. Oper. Res.*, vol. 6, no. 1, pp. 19–30, 1981.

[71] S. Hernández, G. Leguizamón, and E. Mezura-Montes, "Hibridación de evolución diferencial utilizando hill climbing para resolver problemas de optimización con restricciones," in *XVIII Congreso Argentino de Ciencias de la Computación*, pp. 1–10, Oct 2012.

[72] S. Hernandez, G. Leguizamon, and E. Mezura-Montes, "A hybrid version of differential evolution with two differential mutation operators applied by stages," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 2895–2901, June 2013.

[73] S. Dominguez-Isidro, E. Mezura-Montes, and G. Leguizamon, "Memetic differential evolution for constrained numerical optimization problems," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 2996–3003, June 2013.

[74] C. Zhang, J. Chen, and B. Xin, "Distributed memetic differential evolution with the synergy of lamarckian and baldwinian learning," *Appl. Soft Comput.*, vol. 13, pp. 2947–2959, May 2013.

[75] A. P. Piotrowski, "Adaptive memetic differential evolution with global and local neighborhood-based mutation operators," *Inf. Sci.*, vol. 241, pp. 164–194, Aug. 2013.

[76] M. Vakil-Baghmisheh and M. Ahandani, "A differential memetic algorithm," *Artificial Intelligence Review*, vol. 41, no. 1, pp. 129–146, 2014.

[77] B. Liu, H. Ma, X. Zhang, and Y. Zhou, "A memetic co-evolutionary differential evolution algorithm for constrained optimization," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 2996 –3002, sept. 2007.

[78] A. Menchaca-Mendez and C. A. Coello Coello, "A new proposal to hybridize the nelder-mead method to a differential evolution algorithm for constrained optimization," in *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pp. 2598–2605, May.

[79] F. Zhao, M. Huo, and W. Ma, "A Memetic Differential Evolution Algorithm with Adaptive Mutation Operator," *Research Journal of Applied Sciences*, vol. 4, no. 19, pp. 3687–3691, 2012.

[80] M. Pescador Rojas and C. A. Coello Coello, "A memetic algorithm with simplex crossover for solving constrained optimization problems," in *World Automation Congress (WAC), 2012*, pp. 1 –6, june 2012.

[81] X. Pan, "Adaptive Differential Evolution with Local Search for Solving Large-scale Optimization Problems," vol. 2, no. February, pp. 489–496, 2012.

[82] F. Neri, J. Toivanen, G. L. Cascella, and Y. S. Ong, "An adaptive multimeme algorithm for designing hiv multidrug therapies," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, pp. 264–278, April 2007.

[83] F. Neri, J. Toivanen, and R. a. E. Mäkinen, "An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV," *Applied Intelligence*, vol. 27, no. 3, pp. 219–235, 2007.

[84] V. Tirronen, F. Neri, T. Karkkainen, K. Majava, and T. Rossi, *A Memetic Differential Evolution in Filter Design for Defect Detection in Paper Production*, pp. 320–329. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[85] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[86] S. Hu, H. Huang, and D. Czarkowski, "Hybrid trigonometric differential evolution for optimizing harmonic distribution," *Proceedings - IEEE International Symposium on Circuits and Systems*, vol. 1, no. 1, pp. 1306–1309, 2005.

[87] J. Santamaría, O. Cordón, S. Damas, J. M. García-Torres, and A. Quirin, "Performance evaluation of memetic approaches in 3D reconstruction of forensic objects," *Soft Computing*, vol. 13, no. 8-9, pp. 883–904, 2009.

[88] M. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Computer Journal*, vol. 7, pp. 155–162, 1964.

[89] F. J. S. Wets and R. J-B., "Minimization By Random Search Techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, 1981.

[90] J. Leskinen, F. Neri, and P. Neittaanmäki, *Memetic Variation Local Search vs. Life-Time Learning in Electrical Impedance Tomography*, pp. 615–624. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[91] X. Fu and J. Yu, "A hybrid algorithm based on Extremal Optimization with adaptive levy mutation and Differential Evolution and application," *5th International Conference on Natural Computation, ICNC 2009*, vol. 1, pp. 12–16, 2009.

[92] M. Cruz-Ramírez, J. Sánchez-Monedero, F. Fernández-Navarro, J. C. Fernández, and C. Hervás-Martínez, "Hybrid Pareto differential evolutionary artificial neural networks to determined growth multi-classes in predictive microbiology," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6098 LNAI, no. PART 3, pp. 646–655, 2010.

[93] F. Neri and E. Mininno, "Memetic compact differential evolution for cartesian robot control," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 54–65, 2010.

[94] X. Li and M. Yin, "Hybrid differential evolution with artificial bee colony and its application for design of a reconfigurable antenna array with discrete phase shifters," *IET Microwaves, Antennas Propagation*, vol. 6, pp. 1573–1582, November 2012.

[95] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Appl. Soft Comput.*, vol. 8, pp. 687–697, jan 2008.

[96] V. Basetti and A. K. Chandel, "Hybrid power system state estimation using taguchi differential evolution algorithm," *IET Science, Measurement Technology*, vol. 9, no. 4, pp. 449–466, 2015.

[97] J. Ghani, I. Choudhury, and H. Hassan, "Application of taguchi method in the optimization of end milling parameters," *Journal of Materials Processing Technology*, vol. 145, no. 1, pp. 84 – 92, 2004.

[98] S. Elsayed, R. Sarker, and C. A. Coello Coello, "Enhanced Multi-operator Differential Evolution for Constrained Optimization," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*, pp. 4191 – 4198, 2016.

[99] S. Surender Reddy, J. Y. Park, and C. M. Jung, "Optimal operation of microgrid using hybrid differential evolution and harmony search algorithm," *Frontiers in Energy*, 2016.

[100] A. F. Ali, N. N. Ahmed, N. A. M. Sherif, and S. Mersal, *Hybrid differential evolution and simulated annealing algorithm for minimizing molecular potential energy function*, vol. 407 of *Advances in Intelligent Systems and Computing*. 2016.

[101] N. Khan, F. Neri, and S. Ahmadi, "Adaptive differential evolution applied to point matching 2d gis data," in *Proceedings - 2015 IEEE Symposium Series on Computational Intelligence, SSCI 2015*, pp. 1719–1726, 2016.

[102] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical

benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006. Cited By :1475.

[103] F. Neri and N. Khan, "Two local search components that move along the axes for memetic computing frameworks," in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - FOCI 2014: 2014 IEEE Symposium on Foundations of Computational Intelligence, Proceedings*, pp. 62–69, 2015. Cited By :4.

[104] K. Ma, P. Yan, and W. Dai, "A hybrid discrete differential evolution algorithm for dynamic scheduling in robotic cells," in *2016 13th International Conference on Service Systems and Service Management (ICSSSM)*, pp. 1–6, June 2016.

[105] Q.-K. Pan, L. Wang, and B. Qian, "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems," *Comput. Oper. Res.*, vol. 36, pp. 2498–2511, aug 2009.

[106] F. Chauvet, E. Levner, L. K. Meyzin, and J.-M. Proth, "On-line scheduling in a surface treatment system," *European Journal of Operational Research*, vol. 120, no. 2, pp. 382 – 392, 2000.

[107] P. Rakshit, A. Konar, P. Bhowmik, I. Goswami, S. Das, L. C. Jain, and A. K. Nagar, "Realization of an adaptive memetic algorithm using differential evolution and q-learning: A case study in multirobot path planning," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 43, no. 4, pp. 814–831, 2013.

[108] F. Zhong, B. Li, and B. Yuan, "Circuit tolerance design by differential evolution with hybrid analysis method," in *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*, pp. 74–78, Feb 2016.

[109] A. Caponio, F. Neri, G. L. Cascella, and N. Salvatore, "Application of Memetic Differential Evolution frameworks to PMSM drive design," *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, pp. 2113–2120, 2008.

[110] M.-L. Chen and F. S. Wang, "Fuzzy optimization for a batch simultaneous saccharification and co-fermentation process by hybrid differential evolution," in *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8, June 2012.

[111] S. Das, A. Konar, and U. K. Chakraborty, "Annealed differential evolution," in *2007 IEEE Congress on Evolutionary Computation*, pp. 1926–1933, Sept 2007.

[112] C. Y. Chung, C. H. Liang, K. P. Wong, and X. Z. Duan, "Hybrid algorithm of differential evolution and evolutionary programming for optimal reactive power flow," *IET Generation, Transmission Distribution*, vol. 4, pp. 84–93, January 2010.

[113] D. B. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Transactions on Neural Networks*, vol. 5, pp. 3–14, January 1994.

[114] A. Semnani, I. T. Rekanos, M. Kamyab, and T. G. Papadopoulos, "Two-dimensional microwave imaging based on hybrid scatterer representation and differential evolution," *IEEE Transactions on Antennas and Propagation*, vol. 58, pp. 3289–3298, Oct 2010.

[115] N. Duvvuru and K. S. Swarup, "A hybrid interior point assisted differential evolution algorithm for economic dispatch," *IEEE Transactions on Power Systems*, vol. 26, pp. 541–549, May 2011.

[116] A. Parassuram, S. N. Deepa, and M. Karthick, "A hybrid technique using particle swarm optimization and differential evolution to solve economic dispatch problem with valve-point effect," in *2011 INTERNATIONAL CONFERENCE ON RECENT ADVANCEMENTS IN ELECTRICAL, ELECTRONICS AND CONTROL ENGINEERING*, pp. 51–56, Dec 2011.

[117] Y. Fu, M. Ding, C. Zhou, and H. Hu, "Route planning for unmanned aerial vehicle (uav) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, pp. 1451–1465, Nov 2013.

[118] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, pp. 39–43, Oct 1995.

[119] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, vol. 1, pp. 325–331 Vol.1, June 2004.

[120] C. Jena, M. Basu, and C. K. Panigrahi, "Differential evolution with Gaussian mutation for combined heat and power economic dispatch," *Soft Computing*, vol. 20, no. 2, pp. 681–688, 2014.

[121] F. Zaman, S. U. Khan, K. Ashraf, and I. M. Qureshi, "An application of hybrid differential evolution to 3-d near field source localization," in *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014*, pp. 474–477, Jan 2014.

[122] A. Forsgren, P. E. Gill, and M. H. Wright, "Interior methods for nonlinear optimization," *SIAM Review*, vol. 44, pp. 525–597, 2002.

[123] T. Ganesan, P. Vasant, I. Elamvazuthi, and K. Z. K. Shaari, "Game-theoretic differential evolution for multiobjective optimization of green sand mould system," *Soft Computing*, pp. 3189–3200, 2015.

[124] C. Wang, Y. Fang, and S. Guo, "Multi-objective optimization of a parallel ankle rehabilitation robot using modified differential evolution algorithm," *Chinese Journal of Mechanical Engineering*, vol. 28, no. 4, pp. 702–715, 2015.

[125] W. Ma, M. Wang, and X. Zhu, "Hybrid particle swarm optimization and differential evolution algorithm for bi-level programming problem and its application to pricing and lot-sizing decisions," *Journal of Intelligent Manufacturing*, vol. 26, no. 3, pp. 471–483, 2015.

[126] K. K. Dhaliwal and J. S. Dhillon, "Integrated Cat Swarm Optimization and Differential Evolution Algorithm for Optimal IIR Filter Design in Multi-Objective Framework," *Circuits, Systems, and Signal Processing*, 2016.

[127] S.-C. Chu, P.-w. Tsai, and J.-S. Pan, *Cat Swarm Optimization*, pp. 854–858. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[128] H. Guo, Y. Li, X. Liu, Y. Li, and H. Sun, "An enhanced self-adaptive differential evolution based on simulated annealing for rule extraction and its application

in recognizing oil reservoir," *Applied Intelligence*, vol. 44, no. 2, pp. 414–436, 2016.

[129] A. Ullah, R. Sarker, D. Cornforth, and C. Lokan, "An agent-based memetic algorithm (ama) for solving constrained optimazation problems," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pp. 999–1006, Sept 2007.

[130] E. Mezura-Montes, M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization: An empirical study," *Inf. Sci.*, vol. 180, pp. 4223–4262, Nov. 2010.

[131] F. Neri, J. Toivanen, G. Cascella, and Y.-S. Ong, "An adaptive multimeme algorithm for designing hiv multidrug therapies," *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, vol. 4, pp. 264–278, April 2007.

[132] R. Mallipeddi and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization," tech. rep., Nanyang Technological University, Singapore, 2010.

[133] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[134] S. Dominguez-Isidro, E. Mezura-Montes, and L. Guillermo, "Evolutionary Programming for the Length Minimization of Addition Chains," *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 125–134, 2015.

[135] J. J. Liang, T. P. Runarsson, M. Clerc, P. N. Suganthan, C. A. Coello Coello, and K. Deb, "Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization," *Evolutionary Computation*, pp. 251–256, 2006.

[136] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[137] D. Jia, G. Zheng, and M. Khurram Khan, "An effective memetic differential evolution algorithm based on chaotic local search," *Information Sciences*, vol. 181, no. 15, pp. 3175–3187, 2011.

[138] M. T. Vakil-Baghmisheh and M. A. Ahandani, "A differential memetic algorithm," *Artificial Intelligence Review*, vol. 41, no. 1, pp. 129–146, 2014.

[139] M. Pescador-Rojas, *Un algoritmo memético para optimizacién de espacios restringidos Miriam Pescador Rojas Maestra en Ciencias Computación*. PhD thesis, Instituto Politecnico Nacional, 2010.

[140] C. Zhang, J. Chen, and B. Xin, "Distributed memetic differential evolution with the synergy of Lamarckian and Baldwinian learning," *Applied Soft Computing*, vol. 13, no. 5, pp. 2947–2959, 2013.

[141] E. Mezura-Montes, M. E. Miranda-Varela, and R. d. C. Gómez-Ramón, "Differential Evolution in Constrained Numerical Optimization. An Empircal Study," *Information Sciences*, vol. 180, pp. 4223–4262, 2010.

[142] A. Draa, S. Bouzoubia, and I. Boukhalfa, "A sinusoidal differential evolution algorithm for numerical optimisation," *Applied Soft Computing*, vol. 27, pp. 99–126, 2015.

[143] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 99–110, 2004.

[144] W. Jakob, "A general cost-benefit-based adaptation framework for multimeme algorithms," *Memetic Computing*, vol. 2, no. 3, pp. 201–218, 2010.

[145] S. Dominguez-Isidro and E. Mezura-Montes, "Study of direct local search operators influence in memetic differential evolution for constrained numerical optimization problems," in *2017 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pp. 1–8, Feb 2017.

[146] V. Tirronen and F. Neri, *Differential Evolution with Fitness Diversity Self-adaptation*, pp. 199–234. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[147] J. Cheng, G. Zhang, F. Caraffini, and F. Neri, "Multicriteria adaptive differential evolution for global numerical optimization," *Integrated Computer-Aided Engineering*, vol. 22, pp. 103–107, apr 2015.

[148] G. Iacca, F. Caraffini, and N. Ferrante, "Multi-strategy coevolving aging particle optimization," *International Journal of Neural Systems*, vol. 24, no. 01, p. 1450008, 2014. PMID: 24344695.

[149] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization problems," *Computers & Operations Research*, vol. 38, no. 12, pp. 1877–1896, 2011.

[150] R. A. Sarker, S. M. Elsayed, and T. Ray, "Differential Evolution With Dynamic Parameters Selection for Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 5, pp. 689–707, 2014.

[151] W. Gong, Z. Cai, and D. Liang, "Differential Evolution for Constrained Optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 4, pp. 716–727, 2015.

[152] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Training and testing a self-adaptive multi-operator evolutionary algorithm for constrained optimization," *Applied Soft Computing Journal*, vol. 26, no. C, pp. 515–522, 2014.