Study of Genetic Algorithm for Optimization Problems

Beatriz Cristina Flamia de Azevedo - 38798

Report made under the orientation of

Profª. Drª Ana Isabel Pinheiro Nunes Pereira

Profª. Drª Glaucia Maria Bressan

Master in Industrial Engineering - Eletric Engineering

2019-2020

Study of Genetic Algorithm for Optimization Problems

Thesis report of

Master Degree in Industrial Engineering - Eletric Engineering

Escola Superior de Tecnologia e Gestão

Federal University of Technology – Paraná

Profª. Drª Ana Isabel Pinheiro Nunes Pereira

Profª. Drª Glaucia Maria Bressan

Beatriz Cristina Flamia de Azevedo - 38798

2019-2020

I certify that I have read this dissertation and that, in my opinion, is appropriate in content and form as a demonstrator of the developed work.

Beatriz Cristina Flamia de Azevedo - 38798

We are like dwarfs on the shoulders of giants, so that we can see more than they, and things at a greater distance, not by any virtue of any sharpness of sight on our part, or any physical distinction, but because we are carried high and raised up by their giant size.

Bernard de Chartres

Acknowledgments

First and foremost, I would like to thank God, for always being by my side protecting and guiding me. I also appreciate the opportunities and challenges provided and all that I have.

I would like to thank my parents Cristina and Narcizo and my grandmother Luiza, for being examples to me, for the incomparable efforts, for supporting and encouraging me even though it is very difficult to be away from those we love.

I also thank my family for all their support, affection and encouragement for my adventures.

I thank some professors who I had throughout the academic trajectory, for making it possible to see more distant. In particular, I wish to express my deep gratitude and sincere thanks to professor Ana Isabel Pereira, for the reception, patience, support and constant cooperation in the development of this work. I heartily thank Glaucia Bressan, for affection, for listening to me and advising me whenever I asked her help. I also thank her for all that I have learned over these years of work and friendship. And I also thank Roberto Molina, for friendship and for the encouraging words that are often fundamental.

I would like to thank my friends that are part of my journey. I appreciate being able to have their support, affection and friendship of all those whom I have the honor of considering my friends. Thanks for everything we have learned from each other, for always having someone to talk, cry and smile. I would also grateful to my friends of "Centro

de Cálculo" who I spent most of my time with this past year, I appreciate all support, affection and especially what we learned in our enriching discussions about absolutely everything.

I appreciate the UTFPR, because this work represents the end of a 10 - year cycle of which I was part of this institution that provided me personal and professional growth and has a great contribution to who I am. I especially thank the projects that I was part of and people that encourage us to seek the best. After all, challenges are meant to be overcome, obstacles surpass and problems solved.

My thanks to IPB, for the structure and opportunity to be part of this institution.

Finally, to all those who I didn't mention but somehow helped me get here, my sincere respect and gratitude.

Abstract

This work consists in to explore the Genetic Algorithms to solve non-linear optimization problems. The aim of this work is to study and develop strategies in order to improve the performance of the Genetic Algorithm that can be applied to solve several optimization problems, as time schedule, costs minimization, among others. For this, the behavior of a traditional Genetic Algorithm was observed and the acquired information was used to propose variations of this algorithm. Thereby, a new approach for the selection operator was proposed, considering the abilities of population individuals to generate offspring. In addition, a Genetic Algorithm that uses dynamic operators rates, controlled by the amplitude and the standard deviation of the population, is also proposed. Together with this algorithm, a new stopping criterion is also proposed. This criterion uses population and the problem information to identify the stopping point. The strategies proposed are validated by twelve benchmark optimization functions, defined in the literature for testing optimization algorithms. The dynamic rate algorithm results were compared with a fixed rate Genetic Algorithm and with the default *Matlab* Genetic Algorithm, and in both cases, the proposed algorithm presented excellent results, for all considered functions, which demonstrates the robustness of the algorithm for solving several optimization problems.

Keywords: Optimization, Genetic Algorithm, Mathematical Modeling.

Resumo

Este trabalho consiste em explorar o Algoritmo Genético para resolução de problemas de otimização não-linear. O objetivo deste trabalho é estudar e desenvolver estratégias para melhorar o desempenho do Algoritmo Genético que possa ser aplicado para resolução de problemas de otimização variados, como escalonamento de horários, minimização de custos, entre outros. Para isso, foi observado o comportamento usual do Algoritmo Genético e as informações adquiridas foram usadas para propor variações deste algoritmo. Assim, uma nova abordagem para o operador de seleção é proposta, considerando a habilidade dos indivíduos da população em gerar descendentes. Além disso, também é proposto um Algoritmo Genético que utiliza taxas dinâmicas nos operadores, controladas pela amplitude e desvio padrão da população. Juntamente com este algoritmo, um novo critério de paragem também é proposto. Este critério utiliza informações da população e do problema de otimização para determinar o local de paragem. As estratégias propostas são validadas por doze funções de teste, definidas na literatura para teste de algoritmos de otimização. Os resultados do algoritmo de taxas dinâmicas foram comparados com um Algoritmo Genético de taxas fixas e com o Algoritmo Genético padrão disponível no *Matlab*, e em ambos os casos o algoritmo proposto apresentou excelentes resultados, para todas as funções consideradas, o que demonstra a robustez do método para resolução de problemas de otimização variados.

Palavras-chave: Otimização, Algoritmos Genéticos, Modelagem Matemática.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Optimization is a mathematics field that studies the identification of functions' extreme points, either maximal or minimal. It is an important tool in the decision making and analysis of systems, currently being used in various areas such as engineering process, medical research, financial analysis and management decision, for example.

The existence of optimization methods can be traced to days of Newton, Lagrange and Cauchy. Newton and Lagrange provided expressive contribution to differential calculus. Later, Bernoulli, Euler and Weirstrass afforded contributions to calculus of variations, which deals with minimization of function, essential to solve optimization problems [1]. However, the optimization methods had their main development in the early period of World War II, when the military called upon a team of mathematicians to develop methods for solving problems of allocating scarce and limited resources (fighter airplanes, radars, submarines) and improve the war strategies [1].

The positive results achieved by the mathematicians had made research in the area of optimization widespread in various commercial and industrial segments after the end of the war. The results of research on (military) operations, subsequently became known as operations research [1]. However, the main advances in operation research techniques were

only possible after the increased processing speed and the amount of computer memory in 20^{th} century [1], [2].

The optimum seeking methods are also known as mathematical programming techniques and they are studied as a part of operations research. The computational advances stimulated studies that make it possible to solve large and complex problems. In addition, the growing interest in optimization has led to discovering new formulations and applications in several science fields.

Inspired by the Darwin natural selection theory, optimization studies also had considerable progress. Scientists observed that nature holds its own optimization mechanisms. Animals, for example, are naturally able to find forms of lower energy expenditure to perform tasks necessary for their survival as reproduction, protection, defense, migration, localization and digestion of food. These natural optimization mechanisms were tightly studied by professor J. H. Holland and his collaborators, that mathematically formalized the process of natural evolution and adaptation of living being [3], [4]. Holland's aim was to improve the understanding of natural adaptation process to design artificial systems having properties similar to natural systems [5]. Although there are no mathematical demonstrations of the optimality of natural optimization mechanisms, there are no doubts that they are essential for species survival and when applied to optimization problem they are able to present excellent results.

The last decade highlighted the use of optimization methods as an essential tool for management, decision making, improvement and development of technologies as this enables competitive advantages to the systems [2]. There are several techniques and algorithms that can be applied to solve optimization problems but there is not an universal algorithm able to solve all problems. Each algorithm is more appropriated to solve a set of problems according to their characteristics. The algorithm choice is very important, since it can determined if it is able to find the exact or approximated solution and how fast this solution will be found. In this work, the Genetic Algorithm, which is a specific optimization

technique is studied with the aim to identify internal patterns and use them to develop strategies to improve the Genetic Algorithm performance and compete with the existing algorithms to solve general optimization problems.

1.1 Background

The idea and necessity of new techniques to solve problems made the optimization important tools, due to several satisfactory results presented by different areas of knowledge. Even though optimization is already used in many areas, it is still a promising field, with a lot of space to be explored.

Many works in the literature study strategies to improve optimization algorithms and consequently improve optimization problem results. Nevertheless, in most works, these improvements are limited to specific applications, not being possible to use them to solve optimization problems in general. Furthermore, to propose algorithms to solve general optimization problems is considered a hard challenge.

The motivation of this work is to explore the Genetic Algorithm and to create variations of the Genetic Algorithm to be used in general optimization problems. This study is dedicated to finding strategies to improve the genetic operators' procedures, aiming to accelerate the population convergence without loss of quality of the optimum solution. For this, the genetic operators' behavior and the population amplitude and standard deviation of the traditional Genetic Algorithm will be observed in order to use the patterns to propose a variation of the Genetic Algorithm based on the population performance throughout the evolutionary process. So, a new approach to select individuals to crossover and mutation, considering the ancestors' ability to generate offspring, is proposed. Moreover, a Genetic Algorithm with dynamic operator rates is also proposed in conjunction with a new stopping criterion. The new algorithm considers three phases in the evolutionary process, in which the operators rates are dynamically adjusted by the amplitude and the standard

deviation of the current population. In the end, the methodologies proposed are validated by twelve benchmark functions and compared with other Genetic Algorithm approaches.

1.2 Objectives

The general objective of this work is to propose some strategies to improve the Genetic Algorithm performance, using information extracted from the traditional Genetic Algorithm, in order to solve general optimization problems. In addition, the specific objectives are listed below:

- To study the characteristics and patterns of Genetic Algorithm, especially of the three main genetic operators: selection, crossover and mutation.
- To study mathematical and statistical strategies to analyse the information present in the algorithms.
- To use the patterns observed to develop variations of the traditional algorithm, that will be competitive with existing algorithms.
- To propose new procedures for select individuals to accelerate the algorithm convergence.
- To propose a Genetic Algorithm with dynamic operator rates.
- To validate the strategies using benchmark optimization functions defined in the literature.
- To compare the results with other Genetic Algorithms approaches.

1.3 Report Organization

This work is structured in 6 chapters and 2 appendixes. In the second chapter, followed by this introduction, are introduced the concepts of mathematical optimization modeling. Besides, the Genetic Algorithm procedures are described. The details of genetic operators (Selection, Crossover and Mutation) and the most useful procedures of these operators are also presented in this chapter.

In the third chapter are presented some studies and variations of the Genetic Algorithm. Different approaches of genetic operators are introduced and compared with the traditional approach.

In the fourth chapter are presented the algorithms and strategies proposed in this work. First, the traditional Genetic Algorithm is described and an analysis of operators' behavior, population amplitude and standard deviation are done. After that, the approaches for a new selector operator are presented. Thereafter, a proposal of a Genetic Algorithm with dynamic operators rates and a new stopping criterion are described.

In the fifth chapter are presented the results obtained and its discussion to taking into account the proposed methods. Besides, some comparisons are done with the traditional Genetic Algorithm from Matlab and a comparison involved the Hybrid Genetic Algorithm is also presented.

The conclusion and suggestion for future work are presented in the sixth chapter.

Finally, the Appendix A presents an example of the Offspring Quality Matrix proposed in chapter four, while the Appendix B describes the benchmark functions used to test and to validate the algorithm variations presented in this work.

Chapter 2

Optimization Problems and Genetic Algorithm

In this chapter are presented the concepts of mathematical optimization modeling in the Section 2.1 and the main difference between the optimization methods in the Section 2.2. Besides, the Genetic Algorithm is detailed in Section 2.3, as well as the genetic operators procedures, in the posterior sections.

2.1 Mathematical Modeling

Mathematical modeling has the mission to describe a real world problem into a mathematical problem, allowing the problems to be solved by algorithms or other computational resources, in a quick and efficient way. An optimization problem can be expressed by a mathematical function named objective function and some algebraic equations and inequalities. The objective function express the objective of the problem, that can be any quantity or combination of quantities that may be represented by a single number as people, time, materials, energy, etc; while the limitations (constraints), that directly affect

the decision and results, are represented by equations and inequalities [1], [6], [7]. The main goal of optimization problems is to minimize, or to maximize, the objective function considering the constraints. Mathematically, an optimization problem can be expressed as:

$$\begin{aligned} \min_{x \in \mathbb{R}^{Nvar}} \quad & f(x), \\ s.t. \quad & g_i(x) = 0, \quad i \in \{1, 2, ..., n_g\} \\ & h_j(x) \geq 0, \quad j \in \{1, 2, ..., n_h\} \\ & {x_t}^L \leq x_t \leq {x_t}^U, \quad t \in \{1, 2, ..., N_{var}\} \end{aligned} \tag{2.1}$$

where $x = (x_1, x_2, ..., x_{Nvar})$ is the variable vector, $f(x)$ is the objective function, that is, a function of x that needs to be minimized. The constraints are described by $g_i(x)$ and $h_j(x)$, and the lower and upper bounds are identified by ${x_t}^L$ and ${x_t}^U$, respectively. The N_{var} is the number of variables, n_g is the number of equality constraints and n_h is the number of inequality constraints [6]. If a given point satisfy all the constraints, then it is named feasible solution.

It is important to highlight that a point x which corresponds to the minimum value of function $f(x)$ also corresponds to the maximum value of -$f(x)$, as illustrated in Figure 2.1 [1].

Solve an optimization problem is hardly ever based on simple mathematical calculations, so due to mathematical complexity, most optimization problems use computational resources to find the best solution. Although there is no single method available for solving all optimization problems efficiently [1], there are various techniques and algorithms may be used to solve optimization problems.

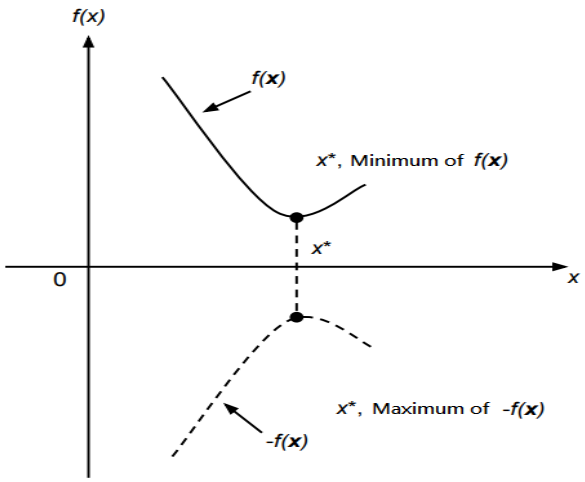

Figure 2.1: Minimum of $f(x)$ is Same as Maximum of $-f(x)$
[1]

2.2 Deterministic versus Stochastic Optimization Methods

According to their characteristics, the mathematical methods used to solve optimization problems can be classified into two large categories: deterministic and stochastic methods.

Deterministic methods are based on systematic expressions and theorems that determine the problem solution, whenever it exists. A method is considered deterministic if it is possible to predict all its steps and the same optimum solution is provided in every time that the algorithm starts, under the same initial conditions. Some deterministic methods use derivative calculation, for this reason, is required an objective function continuous and differentiable in all its domain [1], [6], [7]. Deterministic methods search the entire space of feasible solutions and guarantee the optimally [6], [8], but in some cases, it demands high computational cost and search time, which makes these methods unable for solving non-polynomial time problems (NP-hard). According to [9], deterministic methods present excellent results when the search space is convex, continuous and frictionless, but when these conditions are not guaranteed deterministic methods do not work so well.

Some examples of deterministic methods can be: simplex method, sequential quadratic programming, gradient reducted, branch and bound, among others [5], [6], [8].

Stochastic methods consist of analyzing and transitioning probabilistic rules and use random values in its procedures [5]. The stochastic methods are not always able to find the optimum solution and when it is possible, the optimum solution is slightly different in each algorithm execution. Thus, stochastic optimization methods use the quantification of the uncertainty to produce solutions that optimize the problem [7]. Stochastic methods perform a simultaneous search in the space of possible solutions using a population of individuals, candidates to the optimum solution. These methods do not guarantee the exact or the best solution, as occur in deterministic methods, however, most of the time they present a satisfactory solution, very close to the best solution, in a shorter time than the deterministic techniques. Simulated Annealing, Artificial Immune Systems, Particle Swarm Optimization and Genetic Algorithm are some examples of stochastic methods [5].

A particular class of stochastic methods is the evolutionary algorithms, that use evolutionary strategies to find the solutions of the problems. These algorithms work with many points in the space of solutions and use operators to generate new points and search the optimal solution [10]. The Genetic Algorithm, which is the focus of this work, is one of the most famous examples of evolutionary methods and it will be presented bellow.

2.3 Genetic Algorithm

The Genetic Algorithm (GA) is an optimization method based on the principles of genetics and natural selection [10]. GA is used in large fields, such as image processing, pattern recognition, financial analysis, industry optimization, etc. Ghaheri et. al [11] presents a review of important studies involving Genetic Algorithms in medical specialties. Fera et. al [12] uses Genetic Algorithm to solve the sequential optimization problem in machine with additive manufacturing technology. Alves et. al [13] uses Genetic Algorithm to solve

a healthcare center problem in scheduling home care visits. Li et. al [14] proposes a financial management information system based on improved Genetic Algorithm. Patil and Bhalchandra [15] apply Genetic Algorithm to develop a pattern recognition system to extract features and identify two classes of patterns in two dimensional data space. In all examples cited, Genetic Algorithms showed satisfactory performance, highlighting the good performance in NP-hard problems.

A Genetic Algorithm is composed by a set of individuals, usually named as chromosomes, that are considered solutions for the problem. This set of individuals is known as population, that has a fixed number of individuals in each generation (iteration). The population is represented by N_{pop} individuals arranged in the search space, which is the space where each variable can have values, (some examples are $\mathbb{Z}^{N_{var}}$, $\mathbb{R}^{N_{var}}$, $\{0,1\}^{N_{var}}$, ...). The search space is delimited by the domain of the objective function, this way ensures that all individuals are admitted solution for the problem. Therefore, each possible solution has its objective function value, depending on the problem definition [5].

Individuals can be represented by two forms: binary string or continuous variable in which both represent the same solution. In the binary representation, the individuals work with bits (0 or 1), where the bits can represent a decimal integer, quantitative or qualitative values [10]. It is the most popular method because the binary alphabet offers the maximum number of schemata per bit compared to other coding techniques [4]. On the other hand, in continuous representation, the variables are expressed by floating-points numbers over whatever range is deemed appropriated [10]. The continuous GA has the advantage of requiring less storage than the binary GA because a single floating-point number represents the variable instead of N_{bits} integer [10].

Consider an individual with N_{var} variables (an N_{var} - dimensional optimization problem), then the individual can be represented by a vector $x = (x_1,x_2,...,x_{N_{var}})$, that is presented in Figure 2.2:

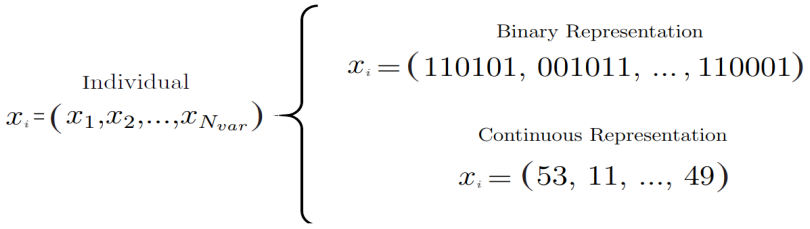

Figure 2.2: Example of Individuals Representations

where each x_i represents an individual gene that is the basic unit of heredity and carries the characteristics of previous individuals. In binary representation the individuals are allocated in a $N_{pop} \times N_{bits}$ matrix, while in continuous representation each row in the matrix is a $1 \times N_{var}$ array individual of continuous values [10].

The basic idea of GA is to create an initial population, $\mathcal{P}^0$, of feasible solutions and dimension N_{pop}, to evaluate each individual, thereafter to select some individuals to define the optimum subset of individuals, with dimension N_{keep}, and to modify them by using genetic operators in each generation k, in order to create new individuals (offspring).

The individuals' evaluation is done by the objective function. The value provided by this function is named objective function value or simply fitness value, which defines how well an individual is adapted to solve the optimization problem. The most adapted individuals have a greater chance of surviving and to generate offspring, while the less adapted are eliminated; similar to what is proposed by Darwin's theory. In other words, the individuals that present better fitness are kept to form the next population. For minimization problem, better fitness is defined as the values which most minimize the objective function, while for maximization problem, better fitness is those which most maximize the objective function. In this work, only minimization problems are considered.

Each population has N_{pop} individuals, where the most adapted or those whose fitness are better will define the N_{keep} individuals and the new population. The individuals of the subset N_{keep} will be selected to generate new individuals or offspring through the genetic operator procedure. The genetic operators are responsible for new individuals creation,

diversification and maintenance of adaptation characteristics acquired in previous generations. The number of individuals used by each operator is defined by the operator rate, that is a fixed value for all iterative process.

In this work, the traditional GA is considered. For this reason, only the three most useful operators (selection, crossover and mutation) are used in the algorithm. Other genetic operators can be consulted in [5]. After performing the operators procedures, the population is evaluated by the objective function and ordered by the fitness value to be selected again. This is an iterative process that ends when a satisfactory solution is found or when a stopping criterion is reached [5], [10]. The codification of a basic Genetic Algorithm is represented by the Algorithm 1 and the main components of GA, initial population $\mathcal{P}^0$, objective function and genetic operators of selection $\mathcal{P}^{'}$, crossover $\mathcal{P}^{''}$ and mutation $\mathcal{P}^{'''}$ are described as follows.

Algorithm 1 : Traditional Genetic Algorithm

Generates a randomly population of individuals, $\mathcal{P}^0$, with dimension N_{pop}.
Set $k = 0$.
while stopping criterion is not met **do**
    $\mathcal{P}' =$ Apply selection procedure in N_{pop} individuals.
    $\mathcal{P}'' =$ Apply crossover procedure in N_{keep} individuals.
    $\mathcal{P}''' =$ Apply mutation procedure in N_{keep} individuals.
    $\mathcal{P}^{k+1} = N_{pop}$ best individuals of $\{\mathcal{P}^k \cup \mathcal{P}'' \cup \mathcal{P}'''\}$.
    Set $k = k + 1$.

2.3.1 Initial Population

The initial population, $\mathcal{P}^0$, can be generated by two ways. The first one consists of using randomly produced solutions created by random numbers. This method is preferred when prior knowledge is not available. However, there are cases in which are possible to use some previous knowledge and requirements about the problem, therefore, is advantageous to use prior information to improve convergence speed, due to the reduction of search space [4]. The size is an important parameter in the initial population, since it affects

the global performance and the efficiency of the Genetic Algorithm. A small population covers a small search space, whereas a big population provides a representative coverage of the problem and prevents premature convergence in local points [4]. It is important to highlight that a larger population requires larger computational resources and demands higher time consuming.

2.3.2 Objective Function

The objective function or fitness function is the way that GA uses to communicate with the optimization problem. The evaluation of the solution quality is provided according to the information produced by this function and not by using direct information about the GA structure [4]. The quality of a proposed solution is usually calculated depending on how well the solution satisfies the given constraints [4]. This value is used to rank individuals depending on their relative suitability for the optimization problem. The complexity of the objective function value depends on the optimization problem. In some cases the mathematical equation cannot be formulated, as result, a rule-based procedure can be constructed for use as a fitness function, or both can be combined [4], [5].

2.3.3 Selection Procedure

The aim of the selection procedure, $\mathcal{P}'$, is to select individuals of the existent population in order to reproduce more copies of individuals (offspring). This procedure is based on the survival probability that depends on the objective function value of the individual. In minimization problems, individuals with low fitness are more likely to survive and get mating rights and individuals with high fitness are easily eliminated [4], [16]. The opposite occurs for maximization problems, in which the higher fitness of individuals are preferable to be kept.

The selection procedure has the mission to guide the algorithm for promising areas, where

the probability to find the best solution is higher. However, the diversity of the population must be maintained to avoid premature convergence and to reach the global optimal solution [4]. This procedure is identical to binary or continuous individuals.

Considering a minimization problem, first, the N_{pop} individuals are ranked from lowest to highest fitness. Then, only the best individuals, in this case, are those whose fitness are smaller, are selected to continue in the population, following the Darwin principle, in which only the most adapted individuals survive.

The number of individuals that are kept and form the subset with N_{keep} individuals at each generation is

$$N_{keep} = r_s \times N_{pop}, \tag{2.2}$$

where r_s means the natural selection rate of N_{pop} individuals that continuous for the next steps in the algorithm [10]. If this rate is 1 all elements of the population N_{pop} are likely to be selected to become parents, on the other hand, if this value is less then 1, only a percentage of individuals is selected to form the population to keep (with dimension N_{keep}) and posteriorly selected to produce new offspring.

There are different ways to selected individuals to become parents, such as proportional selection, ranking-based selection, tournament selection and elitist selection [4], [5], [15]. The most useful selections procedures are described below.

Proportional Selection

Proportional selection is usually known as Roulette Wheel Selection, and it is one of the traditional GA selection techniques [5]. The principle of roulette selection is a linear search through a roulette wheel with the slots in the wheel weighted in proportion to the individual's objective function value. The expected value p_i of an individual is its

objective function value f_i divided by the actual objective function value of the population, mathematically expressed as:

$$p_i = \frac{f_i}{\sum_{j=1}^{N_{keep}} f_i}. \tag{2.3}$$

Each individual is assigned to a slice of the roulette wheel, where the size of the slice is proportional to the individual's fitness. The wheel is spun N_{keep} times. On each spin, the individual under the wheel's marker is selected to be one of the parents for the next generation [5].

Rank Selection

Rank selection is used when the individuals' fitness values in a population differ greatly between them. This selection procedure ranks the population based on their fitness, varying from 1 to N_{keep} times, (number of individuals in the population N_{keep}). Potential parents are selected and a tournament is held to decide which individuals will be the parents. There are many ways this can be achieved and one suggestion could be the random selection, in which the individual with the highest evaluation becomes a parent and the same process is repeated to find a second parent [5].

Tournament Selection

Tournament Selection strategy performs a tournament competition among N_{keep} individuals using the Roulette selection to produce a subset of individuals. The tournament winner is the individual with the lowest fitness for minimization problems or the highest fitness for maximization problems. So, the winner is copied to the new population. This process is repeated until the size reaches to the population size [5], [17].

Elitism Selection

In the Elitism procedure, the first best individuals or the few best individuals are copied to the new population. Elitism is the procedure by which the weakest individual of the current population is replaced by the fittest individual of the immediately preceding population [5].

2.3.4 Crossover Procedure

Crossover denoted by $\mathcal{P}''$, is considered a primary operation, due to the survey of information that is accessible through the search space, which inadvertently improves the behavior of the GA [18]–[20]. This process is used to create new individuals (offspring) from two existing individuals (parents) selected the subset N_{keep}. The crossover is responsible for recombining the parents' characteristic during the reproduction process, so this process is possible for the next generation to inherit the characteristics of their parents. The recombination of the good characteristics of each ancestor can, but not ever, produce "best fit" offspring whose fitness is greater than its parents [5]. For both binary or continuous individuals representations, crossover procedures are very similar, the main difference is the type of variable. Some common crossover operations are one-point crossover, two-point crossover and uniform crossover and they are described below. However, there are several techniques to combine individuals, available in the literature, some of them can be seen in [2], [4], [5], [10].

One-point Crossover

The traditional GA uses one-point crossover [5]. In this procedure, a crossover point, which is a point between the first and the last genes is randomly selected. Then, the two mating individuals are cut once at corresponding points and their sections are exchanged

forming two new offspring [5], [21], as represented in Figure 2.3 (a).

Two-point Crossover

In two-point crossover, two crossover points are randomly selected. Similarly one-point crossover, the individuals are divided, but in this case in three segments, and the exchange between the individuals segments is done [5], [21], as shown in Figure 2.3 (b).

Uniform Crossover

This procedure follows a binary mask, randomly generated for each pair of parents. This mask has the same length as the individuals' parents. Thereby, each gene in the offspring is created by copying the corresponding gene from one or the other parent determined by the mask. As illustrated in the Figure 2.3 (c), where there is a number 1 in the crossover mask, the gene is copied from the first parent, and where there is a number 0 in the mask the gene is copied from the second parent, for the offspring 1. And for the offspring 2, the procedure is the same, but the parents positions are changed, to generated a second offspring diffent from the first one [5].

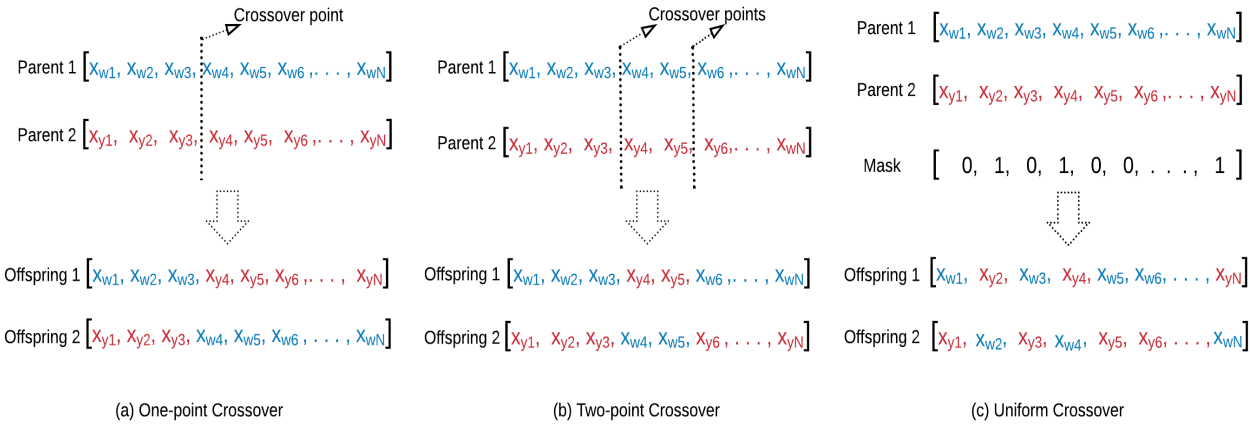

Figure 2.3: Examples of Crossover Operators
Adapted from [5], [10]

2.3.5 Mutation Procedure

The Mutation procedure, $\mathcal{P}''$, is responsible for diversifying the existing population allowing the search by solution in promising areas and avoiding premature convergence in local points [4], [5]. In the basic GA, a mutation probability rate is fixed, for all generations, and it is calculated as a percentage of the N_{keep} individuals that will be mutated in each generation. If the mutation probability is equal to 0 there is no mutation, so, the offspring are generated immediately after crossover (or directly copied) without any change. On the other hand, if the mutation probability rate is different of 0, some elements of the population will have mutation procedure [5]. This process helps the algorithm to escape from local minimum region because it slightly modifies the search direction and introduces new genetic structures in the population. However, the mutation should not occur very often, because then GA will in fact change to random search [5]. For mutation procedure, there are different rules, described below, depending on the type of individual representation.

Flipping Mutation for Discrete Individuals

In this mutation procedure, one individual is considered and a changing of bit (0 to 1 or 1 to 0) is done according to the individual mutation randomly generated. Figure 2.4 (a) illustrates an example of flipping mutation. When the gene of mutation individual is 1 the corresponding bit in the parent individual is flipped and an offspring individual is produced [5].

Interchanging Mutation for Discrete Individuals

In this procedure, two random positions of the string are chosen and the bits corresponding to those positions are interchanged [5]. An example can be seen in Figure 2.4 (b).

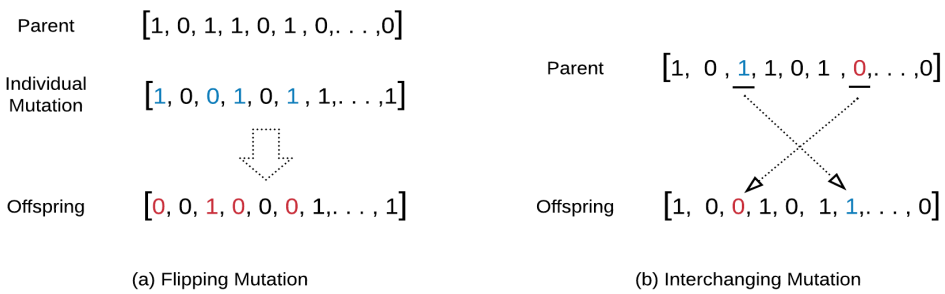

Figure 2.4: Examples of Binary Mutation Operators
Adapted from [5]

Mutation for Continuous Individuals

As occurs in the binary procedure, in continuous GA, a mutation rate is chosen to determine how many individuals of the subset N_{keep} will be mutated. Thereafter a random number is chosen to select variables to be mutated. So, the value of the mutated variable is replaced by a new random value. Example: if the second position is chosen, the value 8.915 is replaced with a uniform number between the function domain, in this case, 3.701, as illustrated in Figure 2.5.

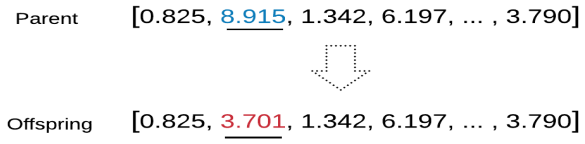

Figure 2.5: Example of Continuous Mutation
Adapted from [10]

2.3.6 Termination Criteria

Iterative optimization algorithm, like GA, requires a stopping criterion to interrupt the search. The criteria choice may affect the quality of the solution found by the algorithm. For example, if the criterion is defined to stop when the first feasible solution is found, the algorithm will not consider that it is possible to have another feasible solution, better than

the first one. Thus, a low quality solution can be chosen wrongly as the best solution. An algorithm can use more than one stopping criterion. Some examples of stopping criteria that can be adopted are described as follow.

Successive Optimum Solution Analysis

This criterion compares the objective function value between two successive populations, using Equation (2.4)

$$|f_k^* - f_{k-1}^*| < \epsilon_1, \tag{2.4}$$

where f_{k-1}^* represents the best solution in the previous population and f_k^* is the best solution of the current population. The algorithm stops when the difference between this value is smaller than a pre-established value, ϵ_1.

Changes in the Population

This criterion compares the population objective function value. As we can see in Equation (2.5), when the difference between f_k^* and $f_k^{\#}$ is smaller than a pre-defined value ϵ_2, the algorithm stops the search. In this case, f_k^* is the minimum objective function value of the generation k and $f_k^{\#}$, is the maximum objective function value of the generation k.

$$|f_k^* - f_k^{\#}| < \epsilon_2, \tag{2.5}$$

Differences Between Successive Values

This criterion compares the differences between successive objective function values into the same population. When this difference is smaller than ϵ_3 the algorithm stops. This

criterion can be represented by Equation (2.6),

$$|f_k^* - f_k^{*'}| < \epsilon_3, \tag{2.6}$$

where $f_k^{*'} = min(\{f_k^1, f_k^2, f_k^3, ..., f_k^{N_{pop}}\} \setminus \{f_k^*\})$.

Optimum Solution Distance

This criterion evaluates the distance between successive optimum solution, as shown in Equation (2.7). If the distance is smaller than ϵ_4 the algorithm stops.

$$||x_k^* - x_{k-1}^*|| < \epsilon_4. \tag{2.7}$$

Number of Generation

A maximum number of generations is defined. Then, if the solution was not found until this number, the algorithm is interrupted.

Time Limit

Time is an important criterion in the Genetic Algorithm, since it can be used to compare the efficiency of the algorithm. In addiction, a remarkable characteristic of Genetic Algorithm is the ability to provide solutions in less time than deterministic methodologies. For these reasons, limited time is defined.

Number of Function Evaluation

Some functions are computationally costly to be evaluated and, for this reason, the maximum number of fitness function evaluation, is defined. So, when the maximum number of evaluations is attained, the algorithm stops.

Chapter 3

Related Works

In this chapter are presented several strategies published, in the last years, in the literature to improve the performance of the Genetic Algorithm. These strategies are compared with the procedures used in the traditional Genetic Algorithm.

3.1 Genetic Algorithm State-of-Art

The performance of GA depends substantially on the efficiency of genetic operators. Several variations of genetic operators are available in the literature and diverse studies have been done to improve these operators [5], [17]–[19], [22], either modifying them or combining them with others methodologies to get optimum solution as early as possible. In some cases, the modifications are proposed to solve exclusive real world problems, in others to compete with different optimization techniques or algorithms.

In the traditional GA, the selection operator is performed among individuals of the same generation. However [17] presents a Back Controlled Selection Operator (BCSO), in which the fitness value of the individual is compared with the fitness value in the previous generation. So, if the fitness value of the individual is more than the one in the preceding

generation, this individual would keep own position, otherwise, would be discarded from the population. The BCSO was compared with six existing operators (roulette wheel selection, sequential selection, tournament selection, dominant selection, hybrid selection, sin selection) in two optimization problems. In the first problem, a Travelling Salesman Problem, the BCSO attained the shortest route compared to the other six methods. And, in the second problem, which is an example of Space Truss Problem, the selection operators were used to achieve the optimum design of the space truss in terms of the weight. In this case, the BCSO gave the minimum weight of steel truss beam and the minimum cost in the pre-stressed precast concrete beam problems. However, the cost of the pre-stressed precast beam, specified for this span, was a little higher than the cost stated by international rules.

Another interesting approach to select individuals can be seen in [21] which utilizes the firefly mating concept to compose a new pair selection operator. The concept utilized by the fireflies algorithms is based on three basic rules: the fireflies are unisex may be attracted to any other fireflies; the level of attractiveness is proportional to the brightness seen by other fireflies; and the brightness is proportional to the value of the objective function. The combination of two individuals with a higher average attractiveness will be chosen for mating. The parameters are evaluated by predetermined functions, defined in [21] and the fireflies' rules are incorporated into GA. A comparison between the traditional GA, the classical fireflies algorithm and the proposed algorithm is presented using four benchmark functions: Sphere function, Rastrigin function, Levy function and Sum Squares function. The results indicated the proposed algorithm is able to find the optimum solution in less time and achieve much higher success than the compared algorithms. Besides, it was possible to note that the results vary widely based on the number of population for the two compared algorithms while this variation was not noted in the proposed algorithm.

Genetic Algorithm and integer programming are combined to solve an example of capacitated p-median problem in [23]. For this, a new crossover operator, that uses the problem

knowledge to select high-quality alleles from the parent's genomes, is proposed. This new crossover preserves only the positive traits of the parents in the offspring. First, the complete genetic information from both parents is collected, producing a vector through the union of parent's genomes. Then, the fitness value of this vector is evaluated, and priorities are assigned to the alleles using some problem knowledge. After this, the alleles are sorted in descending order, according to their priorities, and one offspring is created, containing only p best alleles of the combined vector. Using computational experiments the authors concluded the new crossover method improves the objective function 3.11% on average compared to the original GA with the traditional one-point crossover.

A crossover operator guided by the prior knowledge about the most promising areas in the search space is presented in [24]. Four parameters are defined to control the crossover operator: crossover probability p_c, variable-wise crossover probability p_{cv}, multiplying factor α and directional probability p_d. Initially, $N_{pop}/2$ pairs are formed to be possible parents. A particular pair of solution is allowed to participate in crossover if a random number created between (0,1] is found to be either less than or equal to the crossover probability p_c. The p_c depends on the nature of the objective function. When this condition is satisfied a variable-wise p_{cv} is used to determine the occurrence of the crossover for a certain variable position. The directional information is obtained comparing the mean position of the two mating parents with the position of the current best solution based on a set of equations presented in [24]. When p_d is taken to be equal to 1, the offspring solution follows the current best solution of the population and goes far when p_d is found to be equal to 0. When the p_d is lying between (0,1), the offspring solutions may, or may not, follow the current best solution. The distances among the offspring solutions and the mating parents are controlled by α. It was noted the use of the directional information helps the algorithm to search in more potential regions of the variable space.

A new strategy to solve the traveling salesman problem combining random crossover and dynamic mutation is presented in [16]. To increase population diversity and optimize

mutation characters a random cross mapping method and a dynamic mutation probability are used. In the traditional GA, a crossover point is randomly determined and the crossover section length is half of the original sequence length. In the proposed algorithm the crossover operation varies according to the randomly determined crossover point. If the cross starting point is the sixth element, the result is the same as the one-point crossover. But, if the cross point starts at any location between the second and fifth elements, the result is similar to the partial crossover method with four crossover lengths. On the other hand, if a cross-point starts at any location between the seventh and ninth element, the crossover selection length is less than half of the original sequence length. For the mutation process, the probability that controls the operator is dynamically changed according to population stability. In this case, the mutation probability is inversely proportional to the population stability level. So, when the population is unstable the mutation probability is small, but when the population is almost stable to the local solutions, the mutation probability will increase and be more than the mutation probability of the traditional GA. The performance of the proposed algorithm is compared with the traditional GA and two modified Genetic Algorithm approach, through simulations problem. The results considering convergence speed and the optimal solution, obtained by the proposed algorithm, presented excellent performance.

In order to adjust the mutation and crossover rates, [25] uses the fuzzy inference system to accelerate the attainment of the optimal solution and avoid the stoppage in a local optimum through a synergic effect of mutation and crossover. The referred paper modifies the crossover's and mutation's rate at each generation according to the fuzzy system composed of three inputs: the temporal phase of the search, the trend of the fitness of the candidate solutions and stability of the search process. The inputs and the base rules, provided by experts' knowledge or learning mechanisms, result in two outputs parameters corresponding to the rates of variation for crossover and for mutation, that will be used to control the GA operators. The tests shown in [25] demonstrated the procedure improves the performance of GA, by both speeding up the search and avoiding

premature convergence in a local minimum.

Another remarkable strategy to control the crossover operator is presented by [26] where the properties of Gaussian distribution are used to create a more diversified offspring population. For each parent's gene pair, a Probability Distribution Function (PDF) is defined, in which the value of the gene determines the mean and the variance refers to the confidence of this value, thereby the offspring's PDF is resulted of multiplication of its parent's distributions. A function called *big-sigma* is introduced which follows the uniform distribution to further increase the diversity of children spawned by the same parent pair and controls the algorithm parameters. The comparisons between the GA with and without the devised operator indicated that the operator outperforms the traditional GA for most simple problems. However, the operator also has problems with getting stuck in a local minimum and in those cases, the traditional algorithm version tends to outperform it.

Genetic Algorithm and Bayesian principles are combined in [27], to select the significant features in cells microarray clinical data set. The first algorithm is used to select the significant features in the data set, in this case, the initial population is uniformly initialized, evaluated according to one of these classifiers: k-Nearest Neighbour, Support Vector Machine or Naïve Bayes. The individuals are selected by a tournament with 10% elitism rate and the crossover probability is defined using a specified mathematical expression controlled by the fitness value of the individuals' pair and the maximum average fitness of the population. And, for mutation probability, another expression is used based on the fitness value of the mutated individuals and the median fitness value in the population. The crossover and the mutation operations are repeated until a specific percentage of individuals, set by the user, have fitness value greater than the fitness threshold. If the defined threshold is not reached, a percentage of elite solutions in the current population are mutated and introduced into the new population for the next generation. Probabilities dynamically adapted and operations controlled by threshold value prevent local optimum stoppage and help in achieving better exploration and exploitation in the search space.

The second algorithm is introduced to estimate non-ignorable missing values. In this case, the initial population is initialized according to a Bayesian expression, elaborated for this propose, and the individuals' values are defined by other equations, following Bayesian principles that vary according to the type of data (continuous or discrete). The selection is done using tournament selection and the crossover and mutation rate are also dynamically established as occur in the first algorithm [27].

From the references presented is possible to verify the state-of-art of Genetic Algorithm has a wide range, not only in the theory applicability but also in efforts to improve existing techniques through the use of hybrid methodology, which aims to extract the best of each technique and aggregated in a single model. The successful implementation of GA depends directly on the efficiency of genetic operators. Each way to implement the genetic operator has its own advantages and disadvantages under various circumstances. Thus, the key issue in developing a GA is to provide a balance between the properties of the genetic operators and coding algorithm to produce a good performance as a whole.

Chapter 4

Genetic Algorithm Variants

In this chapter are described the analysis done in the traditional GA and the strategies proposed in this work to improve the GA performance. First, in Section 4.1, the traditional Genetic Algorithm is described. This version is based on [28], which consider the fixed rates for all genetic procedures. Analysis of the behavior of traditional GA is performed to inspire new operators' procedures developments. After that, the modification and strategies proposed in this work are presented in Section 4.2.

4.1 Traditional Genetic Algorithm

In the traditional GA, the initial population is randomly determined through to Equation (4.1), in which x_t^U and x_t^L refer to upper and lower limits, respectively, and τ refers to a random number between (0,1]. This equation ensures that all individuals, x, satisfy the limits of the problem, and can be admitted as a feasible solution

$$x = x_t^L + (x_t^U - x_t^L) \times \tau. \tag{4.1}$$

The individuals generated by Equation (4.1) constitute the $\mathcal{P}^0$ population with dimension N_{pop} individuals. All individuals of the population N_{pop} are evaluated by the objective function and they are ordered according to the objective function value. As already said in Section 2.3, in this work, only minimization problems are considered. Thus, the individuals are order from lowest to uppermost, in all algorithms. After the evaluation, a percentage of individuals are selected according to the natural selection rate, forming the subset with N_{keep} individuals, i.e. $(N_{keep} \in \{1, ..., N_{pop}\})$.

Thereafter, the N_{keep} individuals are randomly chosen to constitute pairs and to perform the crossover. Each pair generates two new individuals by two-point crossover procedure, as described in Section 2.3.4. The N_{keep} individuals chosen is established according to the crossover rate.

After this, some N_{keep} individuals are chosen to be muted following the continuous mutation procedure, as presented in Section 2.3.5. Similar to the crossover procedure, the number of individuals chosen is defined by a mutation rate.

The new individuals generated in the crossover and mutation procedures are joined with individuals of the set N_{pop} and all individuals are evaluated by the objective function and ordered. After this, the N_{pop} most adapted individuals survive to constitute the next generation, while the less adapted are eliminated. These procedures are done until a stopping criterion be achieved.

In the traditional GA, used in this work, it is considered the continuous individuals' representation, the population of $N_{pop} = 100$ individuals, natural selection rate equal to 0.5 and crossover and mutation rates equal to 0.25, as suggested by [28]. All rates are constant values for all search process. Besides, the following stopping criteria are used: Successive optimum solution, considering $\epsilon_1 = 10^{-6}$; the optimum solution distance considering $\epsilon_4 = 10^{-6}$; the maximum number of function evaluation equal to 10000 and the maximum number verification of the same solution, equal to 1000.

4.1.1 Analysis of Operators Behavior

In this section, the offspring quality generated is analyzed in order to identify the ability of individuals to survive in the evolutionary process and generate offspring. For this, a new parameter called Cutfitness (C_f) is inserted into the algorithm. The C_f is the fitness value, or the objective function value, of the less adapted individual of the subset with N_{keep} individuals. The C_f is compared with the objective function value of each offspring and this value is updated at each generation. The offspring whose objective function values are smaller than the C_f are considered high quality offspring.

In order to store the quantity of high quality offspring generated by each individual parent, an Offspring Quality Matrix (OQM) is considered. The OQM is composed of five rows and N_{pop} columns. Each column of the OQM represents an individual of the population P^k, where k represents the iteration. The first row, $f(x_i)$, stores the objective function value of the individual x_i. The second and the third rows store information about the mutation and crossover offspring. In particular, $CU(x_i)$ indicates the number of crossover procedures that the individual participated, and the $HQCU(x_i)$, indicates the number of high quality offspring generated by that individual x_i, in the crossover procedure. The last two rows of OQM store the individual mutation information. So, $MU(x_i)$ is the number of offspring generated by the individual x_i and the $HQMU(x_i)$, stores the number of high quality offspring, i.e. $f(offspring) \leq C_f$. Table 4.1 illustrates a generic example of the OQM. However, in Appendix A is described an example of this matrix.

Table 4.1: Offspring Quality Matrix (OQM)

x_1	x_2	...	$x_{N_{pop}}$
$f(x_1)$	$f(x_2)$	...	$f(x_{N_{pop}})$
$CU(x_1)$	$CU(x_2)$	...	$CU(x_{N_{pop}})$
$HQCU(x_1)$	$HQCU(x_2)$	...	$HQCU(x_{N_{pop}})$
$MU(x_1)$	$MU(x_2)$	...	$MU(x_{N_{pop}})$
$HQMU(x_1)$	$HQMU(x_2)$	...	$HQMU(x_{N_{pop}})$

The OQM conducted to the analysis of the crossover and mutation behavior throughout

the evolutionary process. Figures 4.1 and 4.2 illustrate the accumulated sum per generation, of crossover and mutation procedure, in three different function, Gramacy and Lee - Grlee (dimension 1), Branin (dimension 2) and Rastrigin (dimension 3), for more details about these function, please, consult the Appendix B. Both figures are generated though the accumulated sum of the individuals information that survive until a specific generation. In Figure 4.1 and 4.2 the blue points represent the sum of times that the individuals, of a specific generation, were used in the crossover and mutation procedure, respectively. These values correspond to the sum of the row CU for crossover and the row MU for mutation. These values are equal to the sum of individuals generated at each generation.

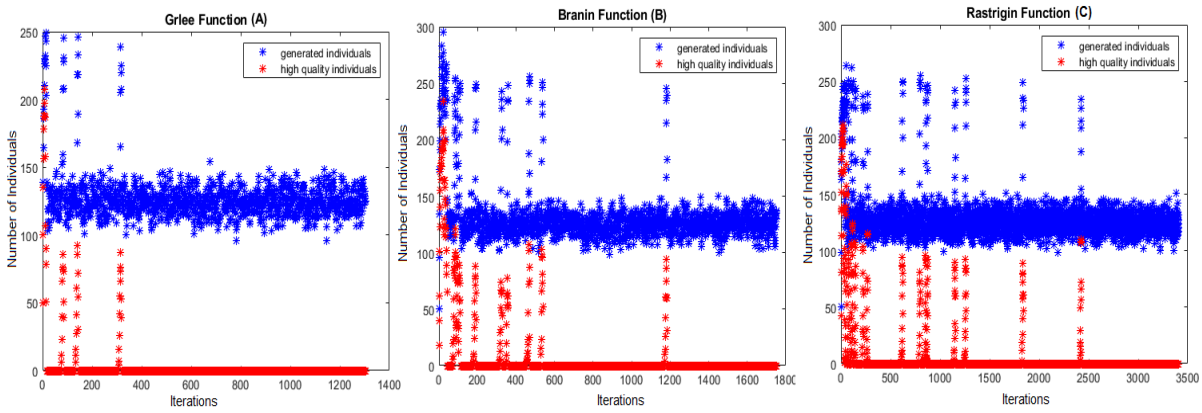

Figure 4.1: Accumulated Value for Crossover Procedure

The red points, in Figures 4.1 and 4.2, represent the accumulated sum of offspring that had the objective function value smaller than the C_f in the crossover and mutation procedure, respectively. In other words, the red points represent the accumulated number of high quality offspring per generation. These points are obtained by the sum of the individuals of the row $HQCU$ for the crossover, and the row $HQMU$ for the mutation procedures.

It is known that new individuals are always being generated while others are eliminated. The aim of the Figures 4.1 and 4.2 is to illustrate the survival of the ancestors and offspring individuals. The more the blue points disperse from the average, it means

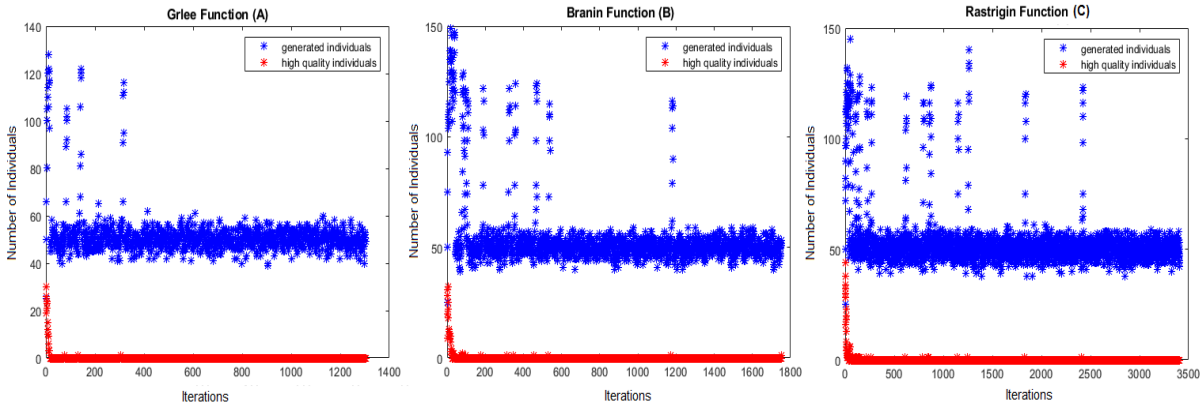

Figure 4.2: Accumulated Value for Mutation Procedure

that there are older living individuals that are used more times in the crossover and the mutation process. The longer individuals survive and generate offspring, the higher are the values stored in the OQM. When the red points have the values near zero, it means that the new population is based on offspring than the ancestors.

From Figures 4.1 and 4.2, it is possible to note that the number of high quality offspring is higher at the beginning than at the end of the search process. The population at the beginning is dispersed and the good solutions have not been found yet, so, high quality offspring are frequently generated. However, in the functions with more complex geometries, such as the Rastrigin function, the number of high quality offspring occurs more throughout the evolutionary process, due to the complexity of the function.

It is noteworthy the fact that there are not so many high quality individuals (red points) in the figures does not mean that the population has not been improved, but the individuals generated did not present an objective function value lower than C_f. As can be seen in Figure 4.3, which ilustrates the average value of the objective function of the population, even with few high quality offspring the objective functions tend to the optimum solution.

From the above mentioned observations, it is intended to develop a new selection operator that gives preference to individuals who generated more high quality offspring and are

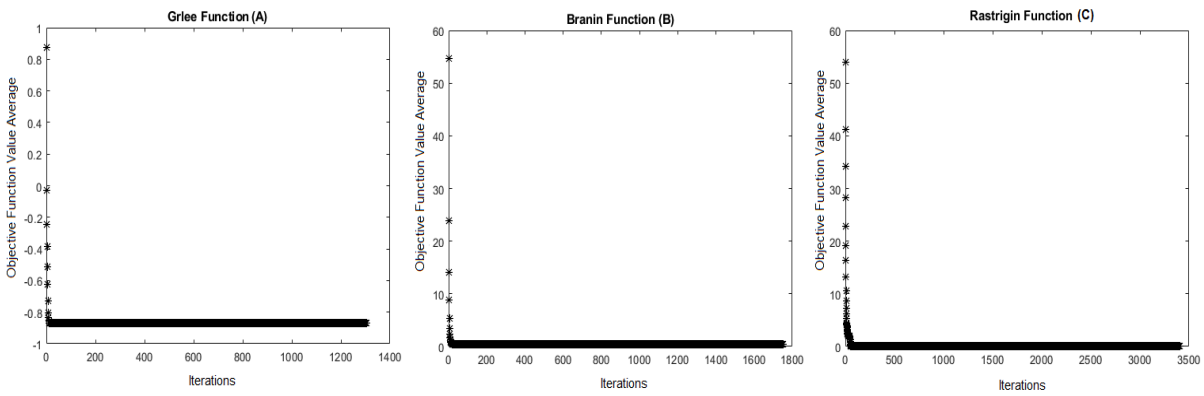

Figure 4.3: Objective Fuction Value Average

still part of the population. For example, consider two individuals which generated, independently, 8 offspring each, one of this individual has 6 high quality offspring, while the other has only 1. So, in this new approach, the individual with more high quality offspring will have a higher possibility to be selected to crossover or mutation procedure.

For less complex functions, this procedure may not work so well, due to the low number of high quality individuals after the initial phase. However, for more complex functions the number of high quality offspring is representative throughout the evolutionary process. Thus, the individuals who have more high quality offspring will have higher possibility to be selected and originate more high quality offspring. Nevertheless, the fact that an individual has not generated any offspring does not eliminate it from the selection process.

4.1.2 Analysis of Amplitude and Standard Deviation

In order to recognize how fast the population change, the population amplitude and standard deviation behavior of the traditional GA were analyzed. Figures 4.4 and 4.5 present, respectively, the amplitude and the standard deviation of the objective function value for three different functions: Grlee Function (dimension 1), Branin Function (dimension 2) and Rastrigin Function (dimension 3). The values of X and Y indicated in Figure 4.4

represent, respectively, an specific iteration and the correspondent amplitude average of this iteration. While in Figure 4.5 the X is also an specific iteration and the Y is the value of the standard deviation average of the iteration that is being considered. More information about these functions can be seen in Appendix B.

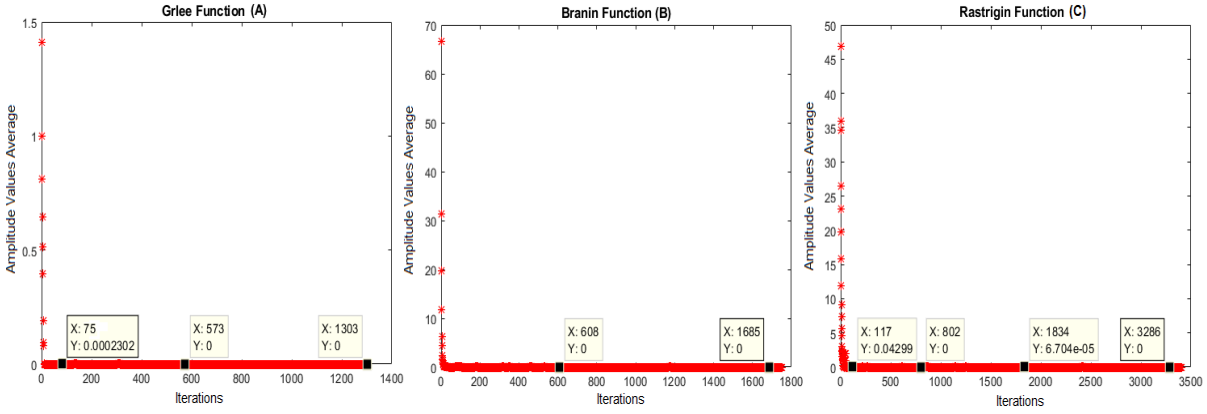

Figure 4.4: Examples of Objective Functions Amplitudes

Observing Figures 4.4 and 4.5, it is possible to conclude that the amplitude and the standard deviation, after the initial iteration, decrease dramatically, since the population tends to converge to an optimum solution point very fast.

In the end of the search, the standard deviation is close to 10^{-15} or 0. However, in simpler functions as Grlee, Figure 4.5 (A), this value can be achieved in less iterations than more complex functions, as Rastrigin, Figure 4.5 (C). For the Grlee function, the algorithm needs several iterations until achieving a stopping criterion. As can be seen in Figure 4.5 (A), in the iteration 75 the standard deviation is near 10^{-15} and it continue near this value until stops, around the iteration 1303. In functions of medium complexity, as Branin function, Figure 4.5 (B), during the 1077 iterations the standard deviation is stable in 10^{-16}. Probably, in this stability period, little modifications were done in the population. On the other hand, in the Rastrigin function, Figure 4.5 (C), more iterations are necessary to find the standard deviation close to 0, due to the function complexity.

In order to facilitate the amplitude and the standard deviation analysis, Figure 4.6

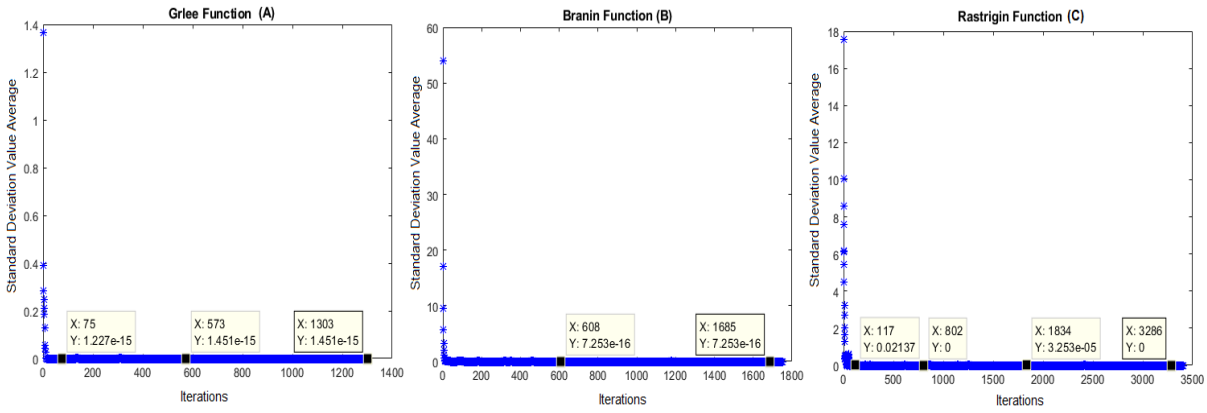

Figure 4.5: Examples of Objective Functions Standard Deviations

presents an approximation of the amplitude and the standard deviation related with Figures 4.4 and 4.5, at the beginning of the search procedures. For all functions considered, at the beginning of the evolutionary process the amplitude and the standard deviation is high. The individuals are dispersed in the search region since the initial population was generated without considering any particular information of the problem. As the process evolves the individuals tend to concentrate in smaller regions (promising regions), where the possibility of finding the best solution are higher. As individuals concentrate in a particular region, the amplitude and the standard deviation of the population decrease. So, the smaller amplitude and standard deviation, more concentrated individuals are around a given region, which has, probably, the optimum solution.

By this analysis, it is possible to conclude that the Genetic Algorithm has different phases in the evolutionary process. At the beginning of the search, the population is dispersed, poor solutions are found and the amplitude and the standard deviation have higher values. In the medium of the process, the individuals tend to concentrate in promising regions, as the solutions are improved, the amplitude and the standard deviation are becoming smaller. At the end of the evolutionary process, the algorithm needs to refine the solutions until converges to an optimum point or attained a stopping criterion. By this observation, a Genetic Algorithm that consider different phases and different operator rates at each

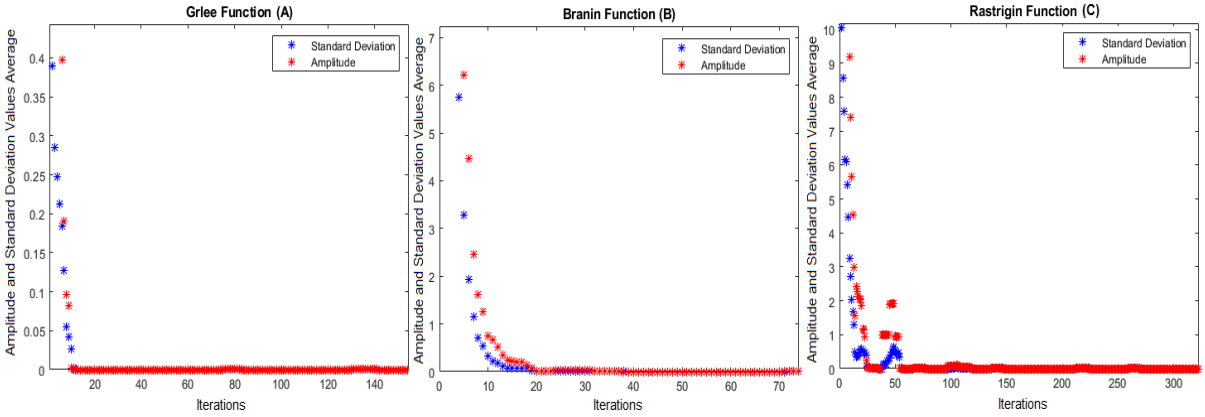

Figure 4.6: Approximations in the Initial Iterations of the Figures 4.4 and 4.5

phase is proposed. Besides, a new stopping criterion based on the amplitude and the standard deviation is proposed. The aim of this criterion is to accelerate the algorithm convergence, without affecting the optimum solution quality.

4.2 Strategies Proposed for Improve the Genetic Algorithm

4.2.1 Conditional Probabilistic Selection

This approach considers the quality of offspring generated in each iteration (generation). In this case, the conditional probability of each individual to generate a high quality offspring is evaluated according to the information stored in the OQM for each individual x_i. As said before, the high quality offspring are those offspring whose objective function values are smaller than the C_f.

In this case, an individual is randomly selected, and the possibility to be accepted, $\tau(x_i)$, to crossover or mutation, is evaluated. This possibility varies between (0,1] and it is evaluated according to three possibilities, described below:

- **The individual selected was never used in crossover, or mutation, procedures, in the previous generation**

  In this case, the individual selected does not have offspring and consequently, none information is available about its ability to generate offspring. For this reason, the possibility to be accepted is maximized, so it is 1. This favors individuals that have never been used more likely to be selected.

- **The individual selected has at least one high quality offspring**

  In this case, an expression based on the metropolis criterion [29] is determined to model the possibility of an individual has to be accepted, according to Equation (4.2) for crossover procedure and Equation (4.3) for mutation procedure

$$\tau(x_i) = exp\left(-1 + \frac{\sum HQCU(x_i)}{\sum CU(x_i)}\right), \tag{4.2}$$

$$\tau(x_i) = exp\left(-1 + \frac{\sum HQMU(x_i)}{\sum MU(x_i)}\right), \tag{4.3}$$

  where x_i represents an individual of the population. The $HQCU(x_i)$ is the sum of high quality offspring generated by the individual x_i in the crossover procedure. And, the $HQMU(x_i)$ is the sum of high quality offspring generated by the individual x_i in the mutation procedure.

- **The individual selected has offspring, but never had a high quality one**

  In this case, the individual selected was used, at least one time in the crossover, or mutation, procedures. So, the accept possibility to be a parent is evaluated using Equations (4.4) and (4.5) for crossover and mutation procedure, respectively.

$$\tau(x_i) = exp\left(-\sum CU(x_i)\right), \tag{4.4}$$

$$\tau(x_i) = exp\left(-\sum MU(x_i)\right), \tag{4.5}$$

where the $CU(x_i)$ and $MU(x_i)$ is the sum of all offspring generation by individual x_i in the crossover and mutation procedure, respectively, in the previous generation.

Thereafter, if $\tau(x_i) > \tau$, where τ is a random number between (0,1], the individual x_i is accepted for crossover or for mutation procedures. The procedure described is repeated until the number of individuals required by the crossover and mutation rates are satisfied.

The main difference between this selection method and the other methods of the literature is the fact that the individual is chosen based on its ability to generate offspring. This promotes individuals that tend to generate high quality offspring are more likely to be selected and consequently accelerate the algorithm convergence to the optimum solution.

4.2.2 Genetic Inheritance Bio-inspired Procedure

The previous results of the new selection operator conducted to studies of how the human genetic code transfer information for the next generation, in order to propose a selection operator inspired by the human genetic information transference.

Some human genetic information are passed through generations, thereby, when a person born his/her genetic characteristics are composed by family genetic information. For example, the eyes or skin color is defined by genetic inheritance.

Thereby, in this approach, it is considered the possibility of the individuals inherit the information of their ancestors. The information considered refers to the offspring quality stored in the OQM. First, it is considered the hypotheses that each new offspring inherit

all information of its family tree (genealogy), i.e., it is considered a new individual that arises in the 20^{th} generation. This individual will inherit the information of the 19^{th} previous generations that compose its genealogy. In this way, the information stored in the OQM for one individual will always be summed and passed to its offspring.

So, the inherited information will be used to calculate the probability of the individual to be accepted for crossover or mutation. Thus, the offspring generated will inherit the ancestors information stored in OQM. And, the random factor that composes the probabilistic approach is understood as the information that not depends on the family information and it is randomly acquired.

The results of this approach proposed will not be presented in this work, due to computer restrictions. This approach requires high computational effort due to the larger quantity of information that should be stored and process in the OQM.

Although a successful result was not obtained by the genetic inheritance bio-inspired propose, the studies carried out so far lead to contributions for potential future work. A possible suggestion to solve the problem found in this approach is to study deeply the human genetic information transference and to determine, for example, how many generations a specify information exerts significant influence on individuals, and thus, consider not all the information of the ancestors, but only the most significant ones. Moreover, it could also assign weights to inherited information. In this way, the information inherited by the previous generation had more influence than the information inherited by more distant generations.

4.2.3 Dynamic Operators Rates

By the analysis of the amplitude and the standard deviation of the traditional GA, it is noted different behaviors in the algorithm evolutionary process, and consequently, it is possible to identify different algorithm search phases.

In the traditional version of GA, the operators have fixed rates for all evolutionary process. The determination of these values is not well established in the literature. Normally, the selection, the crossover and the mutation rates are defined by individual problem analysis or it is determined by means of trial-and-error [30].

There is not a consensus of exact values for each rate, the optimal rate setting is likely to vary for different problems [4], but it is a time consuming task. For these reasons, some studies focused on determining good control rates values for genetic operators. De Jong [31], Schaffer [32] and Grefenstette [33] proposed a range of optimum rates, as presented in Table 4.2, for binary representation individuals [4]. But, as it is possible to see, these ranges have large variations, being inconclusive and strongly dependent on the researcher knowledge and the problem variations.

Table 4.2: Suggested Values for Control Rates of the Genetic Algorithm

Control Rates	De Jong	Schaffer	Grefentette
Population size	50 - 100	20 - 30	30
Crossover rate	0.60	0.75 - 0.95	0.95
Mutation rate	0.001	0.005 - 0.01	0.01

On the other hand, many works in literature are based on the intuitive idea that crossover and mutation rates should not be constant throughout the evolutionary process, but should rather vary in the different phases of the search [25]. Similar to the problem of fixed rates this problem is not simple to solve either. Once again there is no consensus on the values of the rates and how determined them.

There are studies that adapt the control rates during the optimization process. The techniques involve adjusting the operators' rates according to observable problems characteristic as process of the search, trend of the fitness, stability [25], fitness value [4], [27], or based on a number of experiments and expert opinions domain [34].

The work [35] proposes an adaptive mutation through monitoring the homogeneity of the solution population by measuring the Hamming distance between the parents' individuals

during the reproduction. Thereby, the more similar the parents, the higher the mutation probability. The [36] adopted the same mutation probability for all parts of an individual and then decreased to a constant level after a given number of generations. Other strategies are presented in [4], upper and lower limits are chosen for the mutation rate, and within those limits, the mutation rate for each individual is calculated according to its fitness. Some works, for instance, support high mutation rates in the initial phase of the search process, in order to explore the individuals that are spread in the search space [36], [37], while other ones support high mutation rates at the end of the process, in order to come out from local optima, where the algorithm can be stuck [38].

The Dynamic Operators rates approach consists in to control the selection, crossover and mutation rates through the amplitude and the standard deviation of each generation and the objective function value variation. The procedure used to select the individuals are the same used in the Traditional GA, random selection. And the crossover and mutation procedures are also the same as in Traditional GA, to provide a fair comparison, which are two-point crossover and the continue mutation, as described in Section 2.3.4 and Section 2.3.5, respectively.

Through analysis of the traditional GA behavior, it was observed that the objective function standard deviation at the beginning of the process is higher, as expected because the generation of the initial population is random. For this reason, the individuals are very dispersed in the search region. And, for the same reason, the amplitude is also higher at the beginning of the evolutionary process.

As the search process evolves, the population tends to concentrate in specific search region, in which the possibility to find the optimum solution is higher, this causes the decrease of the objective function amplitude and standard deviation. At the end of the evolutionary process both tend to zero, whereas the population is concentrated in a point.

By this analysis is easy to note that the algorithm has different properties throughout the search process. For this reason, this approach proposes to establish different phases and

adapt the operators' rates according to the algorithm answers. Three algorithm search phases were established, as follow described:

- **Phase 1 - Initial**: in this phase, the population amplitude and standard deviation are higher and there are few good solutions in the population. For this reason, higher rates are considered to accelerate the search. This phase initiates when the algorithm starts and stops when the population amplitude and standard deviation smaller than ϵ_i, and a condition of k_i iterations is exceeded.

- **Phase 2 - Development**: this phase starts immediately after *Phase* 1 and ends when the population amplitude and standard deviation are smaller than ϵ_d and exceed at least k_d iterations. The condition of k_d iteration was defined to avoid the algorithm remain too short time at this phase because depending on the problem the ϵ_d value is quickly reached and consequently the possible solutions are not so well explored. In this second phase, the operator rates should not be so higher as in the initial phase because the search is concentrated in the promising areas obtained in the first phase. At the end of this phase, the algorithm has already found the optimum region but needs to refine the optimum solution.

- **Phase 3 - Refinement**: this phase has the aim of refining the solutions. The amplitude and the standard deviation are so small, that it is possible to conclude that the optimum solution is very close of being achieved. The rates used in this phase are smaller than in the other phases because few modifications are needed. This phase starts immediately after *Phase* 2, and continues until a stopping criterion be achieved.

In this approach is not established fixed rates, not even in the phases. Thus, an initial value for each operator rate is determined at the beginning of each phase, as presented in Table 4.3. These values were defined by test using the functions considered in this work.

Table 4.3: Initial Operator Rates for Each Phase

Genetic Operator	*Phase* 1	*Phase* 2	*Phase* 3
Selection	$0.7 \times N_{pop}$	$0.6 \times N_{pop}$	$0.5 \times N_{pop}$
Crossover	$0.5 \times N_{pop}$	$0.4 \times N_{pop}$	$0.3 \times N_{pop}$
Mutation	$0.4 \times N_{pop}$	$0.3 \times N_{pop}$	$0.2 \times N_{pop}$

In the second and the third phases, the initial values rates can vary 10% inside a maximum and minimum interval, according to the difference of amplitude between successive population. Thereby, if the amplitude difference between k and $k-1$ iterations is smaller than ϵ_{ph2} in the second phase or smaller than ϵ_{ph3} in the third phase the rates increase 1%, otherwise decrease 1%. This strategy is inspired by the fuzzy logic principles and it is designed to stimulate small modifications in the algorithm search and prevent it from getting stuck at local points.

After tests and analysis to determine the best values of each phase, which are the good values for all functions considered in this work, it was defined $k_i = 50$, $\epsilon_i = 1$, $k_d = 150$, $\epsilon_d = 10^{-3}$, $\epsilon_{ph2} = 10^{-3}$ and $\epsilon_{ph3} = 10^{-6}$.

The proposed algorithm is shown in Algorithm 2.

Algorithm 2 : Genetic Algorithm with Dynamic Operators Rates

Generates a randomly population of individuals, $\mathcal{P}^0$, with dimension N_{pop}.
Set $k = 0$.
while stopping criterion is not met **do**
    Identify the Phase and update the operator rates if necessary
    $\mathcal{P}' =$ Apply selection procedure in N_{pop} individuals.
    $\mathcal{P}'' =$ Apply crossover procedure in N_{keep} individuals.
    $\mathcal{P}''' =$ Apply mutation procedure in N_{keep} individuals.
    $\mathcal{P}^{k+1} = N_{pop}$ best individuals of $\{\mathcal{P}^k \cup \mathcal{P}'' \cup \mathcal{P}'''\}$.
    Set $k = k + 1$.

4.2.4 Stopping Criterion

In the *Phase* 3, the individuals are near to the optimum solution and few modifications are performed at each iteration in order to preserve the good results already found. In this

way, the following criterion is proposed to be used in conjunction with the other criterion already used in literature, as described in Section 2.3.6.

This stopping criterion consists of an analysis of the population evolution in the *Phase* 3. Therefore, if the algorithm is at *Phase* 3 and the standard deviation and the amplitude are smaller than 10^{-10} in a $N_{pop} \times N_{var}$ successive iterations, the algorithm stops. In this way, the stopping criterion adapts to different problems, since the number of individuals in the population and the variables number are strong influencers in the convergence methods.

The size of the population and the number of variables in an optimization problem have a direct influence on the difficulty that the algorithm has in finding and refining the solution. Functions with simpler geometry and smaller dimensions are easier to be explored by the algorithm, so they need fewer iterations than more complex problems.

By standard deviation observations, it is known that after a certain number of iterations in the *Phase* 3 the population standard deviation is close to 10^{-15}. When a significant population change occurs standard deviation decrease, around 10^{-10}, for this reason, the value of 10^{-10} was chosen to be used in the stopping criterion. Although this value is uncommon in stopping criteria, it presents satisfactory results for all functions considered. It is noteworthy that the value 10^{-10} does not mean that the algorithm will present precision of 10^{-10} in the optimal solution. In addition, this value can be modified if needed.

Chapter 5

Numerical Results and Discussion

In this chapter are presented the results obtained from the methodologies proposed in this work and the comparison with other algorithms. The validation is done by a set of twelve benchmark optimization functions, also presented in this chapter and detailed in Appendix B.

5.1 Benchmark Functions

Benchmark functions are functions compatible with the majority optimization problems [39], [40]. Ideally, these functions should have diverse properties that can be truly useful to test new algorithms in an unbiased way [39]. The validation is only possible if the test suite is large enough to include a wide variety of problems, such as unimodal, multimodal, regular, irregular, separable, non-separable and multi-dimensional problems [39].

Modality refers to the quantity of local optimum the function has. A function with only one local optimum (global optimum) is called unimodal, while the functions with more than one local optimum are called multimodal. Multimodal functions are the most difficult class of problems for many algorithms. This class of functions is used to test the

ability of an algorithm to escape from any local minimum. If the algorithm exploration is poor it cannot search the function landscape effectively and consequently leads to an algorithm getting stuck at a local minimum [39], [41].

Regularity refers to the function surface behavior. Flat, basin and valley, for example, can severely hamper the algorithm search because such regions do not give the algorithm any information to direct the search process towards the minimum [39].

Separability refers to the inter-relation among the function variables. A separable function can be written as a sum of p functions of just one variable while in nonseparable function it is not possible. Separable functions are relatively easy to solve, when compared with their inseparable counterpart, because each function variable is independent of the other variables and can be optimized independently [39].

Dimensionality refers to the number of function dimensions. The difficulty of a problem generally increases with its dimensionality because the search region also increases exponentially [41]. For highly nonlinear problems, this dimensionality may be a significant barrier for almost all optimization algorithms [39].

The benchmark functions could assess convergence speed, accuracy, robustness and their total functionality of optimization and evolutionary algorithms [40]. In this work, twelve benchmark functions, common in literature for optimization algorithms validation, are used: Gramacy and Lee - Grlee (dimension 1) [42], Forrester (dimension 1) [43], Branin (dimension 2) [39], McCormick (dimension 2) [39], Easom (dimension 2)[39], Ackley (dimension 3) [39], [40], Rastrigin (dimension 3) [40], [44], Rosenbrock (dimension 3) [39], Sum Squares (dimension 4) [39], Zakharov (dimension 4) [39], [44], Levy (dimension 5) [44] and Schwefel (dimension 5) [39]. All these functions and their characteristics are described in Appendix B.

The benchmark functions optimum solution defined in the literature are presented in Table 5.1. These optimum solutions will be used to compare the results provided by the

algorithms variations proposed in this work.

Table 5.1: Solution Proposed in Literature

Function	Optimum Solution (f^*)	Function	Optimum Solution (f^*)
Grlee	-0.869011	Rastrigin	0
Forrester	-6.020707	Rosenbrock	0
Branin	0.397887	S. Squares	0
McCormick	-1.913223	Zakhavov	0
Easom	-1	Levy	0
Ackley	0	Schwefel	0

All numerical results of this work were obtained using an Intel(R) Core (TM) i3 CPU M920 @2.67GHz with 6 GB of RAM and the Software *Matlab* R2018a. All the functions are evaluated 100 times and the average of the optimum solution f^*, function evaluations k, time in seconds, and the Euclidean distance between the optimum solution, are considered for the analysis of the results. The Euclidean distance is defined by $||x_i^* - x^*||$, where x_i^* is the solution given by the proposed algorithym and x^* is the solution presented in the Literature.

5.2 Results and Comparision of the Genetic Algorithm with Fixed Rates and the Genetic Algorithm with Conditional Probabilistic Selection

The results obtained using conditional probabilistic selection (GA-CPS), described in the Section 4.2.1, are shown in Table 5.2. The same table also presents the results of the traditional Genetic Algorithm with fixed rates (GA-FR), described in Section 4.1. Both algorithms considered a population of 100 individuals and fixed rates, as suggested by [28], i.e, 0.5 for selection rate and 0.25 for crossover and mutation rates.

The GA-CPS method was able to select the individuals for the crossover and the mutation

Table 5.2: Results of Conditional Probabilistic Selection and Fixed Rates Genetic Algorithms with $N_{pop} = 100$

Function	Algorithm	f^*	k	Function Evaluation	Time	Euclidean Distance
Grlee	GA-FR	-8.690×10^{-1}	1160	60307	2.19	4.009×10^{-5}
	GA-CPS	-8.690×10^{-1}	1215	63178	2.38	3.897×10^{-5}
Forrester	GA-FR	-6.021×10^{0}	825	42887	1.55	3.405×10^{-5}
	GA-CPS	-6.021×10^{0}	900	46841	1.76	3.392×10^{-5}
Brainin	GA-FR	3.979×10^{-1}	1104	57390	2.36	2.791×10^{-3}
	GA-CPS	3.979×10^{-1}	1092	56795	2.38	2.798×10^{-3}
McCormick	GA-FR	-1.913×10^{0}	921	47903	1.97	5.176×10^{-2}
	GA-CPS	-1.913×10^{0}	881	45844	1.96	5.094×10^{-2}
Easom	GA-FR	-9.900×10^{-1}	1203	62547	2.55	1.907×10^{-3}
	GA-CPS	-9.899×10^{-1}	1022	53186	2.18	1.602×10^{-3}
Ackley	GA-FR	8.035×10^{-3}	2700	140423	5.82	3.350×10^{-3}
	GA-CPS	8.156×10^{-3}	2893	150488	6.46	3.560×10^{-3}
Rastrigin	GA-FR	1.006×10^{-4}	2472	128522	5.29	5.960×10^{-4}
	GA-CPS	1.512×10^{-4}	2517	130892	5.56	5.328×10^{-4}
Rosenbrock	GA-FR	4.798×10^{-1}	5444	283085	11.22	1.430×10^{0}
	GA-CPS	4.359×10^{-1}	5032	261675	10.75	1.172×10^{0}
S. Squares	GA-FR	4.929×10^{-5}	1574	81847	3.48	3.450×10^{-3}
	GA-CPS	3.052×10^{-4}	1738	90396	3.89	3.438×10^{-3}
Zakhavov	GA-FR	4.462×10^{-3}	1683	87492	3.71	2.019×10^{-2}
	GA-CPS	5.577×10^{-4}	1972	102580	4.48	1.982×10^{-2}
Levy	GA-FR	4.911×10^{-4}	784	40766	2.14	1.554×10^{-2}
	GA-CPS	4.743×10^{-4}	748	38902	2.08	1.324×10^{-2}
Schwefel	GA-FR	2.121×10^{-3}	3814	198310	8.38	2.121×10^{-3}
	GA-CPS	1.142×10^{-3}	3687	191712	8.41	2.354×10^{-3}

procedures. However, no improvements were found in the proposed approach in relation to the performance of the GA-FR algorithm, which uses traditional procedures. It is noteworthy that the results presented in the Table 5.2 is the average of 100 executions, thus, the small numerical difference between the algorithm results are not representative to affirm that one algorithm is better than the other.

By analyzing the operation methods, it was noted that little information is stored in the OQM, since a high number of new offspring are incorporated at each generation. These new individuals do not have offspring yet, for this reason, it was verified that among the

three conditions proposed in Section 4.2.1, the most used condition was "the individual selected was never used in crossover or mutation procedure, in the previous generation". In this case, the proposed approach is similar to the random selection methods, which is the same used in the GA-FR. Even in more complex functions, which was hoped better performance, due to the higher number of high quality offspring, none significant changes were noted.

5.3 Results and Comparison of Fixed and Dynamic Rates Genetic Algorithms

Table 5.3 presents the results of the Genetic Algorithm with fixed rates ($GA - FR$) and the Genetic Algorithm with dynamic rates ($GA - DR$), considering a population of 100 individuals.

Analyzing the results presented in Table 5.3, both algorithms were able to find the optimum solution with satisfactory precision, in relation to the literature results, presented in Table 5.1.

Although both algorithms, GA-FR and GA-DR, could find the optimum solution with similarity, it is possible to claim that the GA-DR had better performance. The GA-DR used less iteration, time and objective function evaluation than the GA-FR. On average the GA-DR presented 60% fewer iterations and 40% fewer function evaluations than GA-FR. The less computational efforts are required not only by the use of dynamic rates but also for the additional stopping criterion.

In tests done in the GA-DR without the addition of the proposed stopping criterion, demonstrated that the number of iterations k is fewer in GA-DR than the number of GA-FR iterations. But, the number of functions evaluations and the time in GA-DR without the addition of the proposed stopping criterion are equal or little higher than in

Table 5.3: Fixed and Dynamic Rates Genetic Algorithms Results with $N_{pop} = 100$

Function	Algorithm	f^*	k	Function Evaluation	Time	Euclidean Distance
Grlee	GA-FR	-8.690×10^{-1}	1160	60307	2.19	4.009×10^{-5}
	GA-DR	-8.690×10^{-1}	252	23392	0.79	8.718×10^{-5}
Forrester	GA-FR	-6.021×10^{0}	825	42887	1.55	3.405×10^{-5}
	GA-DR	-6.021×10^{0}	241	22460	0.76	5.153×10^{-5}
Branin	GA-FR	3.979×10^{-1}	1104	57390	2.36	2.791×10^{-3}
	GA-DR	3.979×10^{-1}	421	37232	1.42	2.812×10^{-3}
McCormick	GA-FR	-1.913×10^{0}	921	47903	1.97	5.176×10^{-2}
	GA-DR	-1.913×10^{0}	334	30007	1.13	5.197×10^{-2}
Easom	GA-FR	-9.900×10^{-1}	1203	62547	2.55	1.907×10^{-3}
	GA-DR	-9.900×10^{-1}	402	35667	1.37	2.127×10^{-3}
Ackley	GA-FR	8.035×10^{-3}	2700	140423	5.82	3.350×10^{-3}
	GA-DR	1.457×10^{-2}	875	74447	2.85	5.900×10^{-3}
Rastrigin	GA-FR	1.006×10^{-4}	2472	128522	5.29	5.960×10^{-4}
	GA-DR	2.509×10^{-3}	794	67789	2.59	1.532×10^{-3}
Rosenbrock	GA-FR	4.798×10^{-1}	5444	283085	11.22	1.432×10^{0}
	GA-DR	3.971×10^{-1}	2222	184997	6.66	1.160×10^{0}
S. Squares	GA-FR	4.929×10^{-5}	1574	81847	3.48	3.450×10^{-3}
	GA-DR	1.526×10^{-4}	786	67087	2.58	4.686×10^{-3}
Zakhavov	GA-FR	4.462×10^{-3}	1683	87492	3.71	2.019×10^{-2}
	GA-DR	2.733×10^{-4}	1069	89919	3.51	1.042×10^{-2}
Levy	GA-FR	4.911×10^{-4}	784	40766	2.14	1.554×10^{-2}
	GA-DR	2.332×10^{-4}	460	40256	1.93	1.344×10^{-2}
Schwefel	GA-FR	2.121×10^{-3}	3814	198310	8.38	2.121×10^{-3}
	GA-DR	1.570×10^{-3}	2006	167115	6.43	9.703×10^{-2}

GA-FR. For these reasons, the use of the new stopping criterion is recommended in the GA-DR to provide better performance.

The Euclidean distance, which compares the distance between the optimum solution x^*, found by the algorithms and the literature solutions, have the same order of precision. In general, the algorithms solutions are 10^{-3} distance from the literature solution. However, it is worth mentioning that the Euclidean distance of the functions Grlee and Forrester are the smallest, near to 10^{-5}, since they have the less complex geometries and the smallest dimensions. On the other hand, the larger Euclidean distances were presented by Rosenbrock function, which is near to 1, for both algorithms.

5.4 Results and Comparison of Fixed and Dynamic Rates Genetic Algorithms with the Default Matlab Genetic Algorithm

In order to compare the performance of the algorithms using a reduced population, a set of 50 individuals was considered in the results presented in Table 5.4. Besides, the comparison is extended to the default *Matlab* Genetic Algorithm (GA-Mat), defined in [45]. The *Matlab* implementation defines the GA population as 50 individuals when the objective function dimension is less than or equal to 5 or 200 individuals, otherwise.

In Table 5.4 the number of iterations were not considered in the comparison of the algorithms, since the method implemented in *Matlab*, to count the number of iterations, differs from the counting used in this work. Therefore, it is not possible to do a fair comparison.

Table 5.4 shows that GA-FR and GA-DR could provide acceptable optimum solutions for all functions, considering populations of 50 individuals. However, the optimum solutions found by the algorithm with 100 individuals were, in general, a few better. The number of function evaluations and time, were smaller than the results with 100 individuals, due to the reduced size of the population, as presented in Table 5.3.

The results provided by the GA-Mat were not satisfactory for all functions. For the functions Sum Squares, Zakharov and Levy, the GA-Mat presented higher performance than the other algorithms considered, even in relation to the results presented by GA-FR and GA-DR in the population of 100 individuals. However, the GA-Mat presented less performance, in the functions: Grlee, Rastrigin and Schewefel. In the Grlee function, the GA-Mat stopped at a local optimum point, as is clearly observed in the function representation, describe in Appendix B. The worst performance was observed in Schwefel function, in which the GA-Mat could not approximate to the optimum solution, and the

Table 5.4: Default *Matlab*, Fixed and Dynamic Rates Genetic Algorithms Results with $N_{pop} = 50$

Function	Algorithm	f*	Function Evaluation	Time	Euclidean Distance
Grlee	GA-Mat	-6.662×10^{-1}	3250	0.45	9.332×10^{0}
	GA-FR	-8.690×10^{-1}	40522	1.34	5.796×10^{-5}
	GA-DR	-8.690×10^{-1}	10115	0.29	2.169×10^{-4}
Forrester	GA-Mat	-6.021×10^{0}	3300	0.42	3.101×10^{-5}
	GA-FR	-6.021×10^{0}	34914	1.15	9.198×10^{-5}
	GA-DR	-6.021×10^{0}	9656	0.27	9.600×10^{-5}
Branin	GA-Mat	3.979×10^{-1}	3682	0.48	8.689×10^{-4}
	GA-FR	3.979×10^{-1}	45574	1.67	2.523×10^{-3}
	GA-DR	3.979×10^{-1}	14304	0.48	6.170×10^{-3}
McCormick	GA-Mat	-1.913×10^{0}	3500	0.47	1.799×10^{-5}
	GA-FR	-1.913×10^{0}	35915	1.32	5.228×10^{-2}
	GA-DR	-1.913×10^{0}	14323	0.47	5.313×10^{-2}
Easom	GA-Mat	-9.400×10^{-1}	3602	0.49	1.559×10^{-1}
	GA-FR	-9.500×10^{-1}	44218	1.65	1.473×10^{-1}
	GA-DR	-9.421×10^{-1}	12626	0.41	1.804×10^{-1}
Ackley	GA-Mat	2.112×10^{-2}	6065	0.59	9.358×10^{-2}
	GA-FR	1.578×10^{-2}	78323	2.95	6.344×10^{-3}
	GA-DR	6.327×10^{-2}	20827	0.70	2.197×10^{-2}
Rastrigin	GA-Mat	5.771×10^{-1}	4786	0.53	4.836×10^{-1}
	GA-FR	2.094×10^{-4}	76326	2.82	8.610×10^{-4}
	GA-DR	4.142×10^{-3}	20362	0.67	3.625×10^{-3}
Rosenbrock	GA-Mat	9.174×10^{-1}	14526	0.87	1.863×10^{0}
	GA-FR	7.307×10^{-1}	118358	4.29	1.775×10^{0}
	GA-DR	6.296×10^{-1}	23467	0.75	1.492×10^{0}
S. Squares	GA-Mat	6.695×10^{-9}	4885	0.53	4.745×10^{-5}
	GA-FR	1.581×10^{-4}	64222	2.41	5.035×10^{-3}
	GA-DR	9.120×10^{-4}	28142	0.92	1.023×10^{-2}
Zakhavov	GA-Mat	1.937×10^{-8}	5327	0.53	9.186×10^{-5}
	GA-FR	2.726×10^{-4}	80365	3.04	1.066×10^{-2}
	GA-DR	6.394×10^{-4}	40185	1.34	2.025×10^{-2}
Levy	GA-Mat	2.696×10^{-8}	5215	0.55	1.848×10^{-4}
	GA-FR	7.425×10^{-4}	38926	1.80	1.836×10^{-2}
	GA-DR	2.719×10^{-4}	28494	1.18	1.550×10^{-2}
Schwefel	GA-Mat	$1.412 \times 10^{+2}$	9375	0.69	$5.553 \times 10^{+2}$
	GA-FR	4.349×10^{-3}	102264	3.92	1.590×10^{-1}
	GA-DR	2.603×10^{-2}	43663	1.45	3.986×10^{-1}

Euclidean distance is in order of 10^{+2}. For the functions Forrester, Branin, McCormick, Eason, Ackley and Rosenbrock the optimum solutions found, by the three algorithms, were very similar.

In relation to the Euclidean distance, the function which presented the optimum solution with high precision had the smallest Euclidean distance. Regarding the number of function evaluations, GA-Mat has, in general, the lowest values in relation to the GA-FR and GA-DR. And, in relation to the time, the GA-Mat was the fastest in 6 functions. However, the time difference is not so expressive, meanly in relation to the GA-DR time.

Although the GA-Mat presented excellent results for some functions, in other function this algorithm does not work so well. For this reason, the Genetic Algorithm with dynamic rates is more robust and indicated to use in general optimization problems. The GA-DR was able to approximate the optimum solution in all functions considered, with satisfactory precision.

In relation to the use of fewer individuals in the population, it is possible to conclude the Genetic Algorithms performance, considering 50 individuals, were good. But, the use of 100 individuals is more recommended, due to the optimum solution with higher quality. Although the use of 100 individuals demands more time and function evaluations, the difference was not so significant, in the functions considered in this work.

5.5 Results and Comparison with the Hybrid Genetic Algorithm

Some Genetic Algorithms are combined with a local search method to present the global solution with high precision. These algorithms are considered Hybrid Genetic Algorithms. So, in order to improve the results already presented, in this work the Nelder Mead method [46], is used. Previous results showed, that the population of 100 individuals had better

performance. So, in this hybrid approach, 100 individuals were also considered.

It is noteworthy that in the work [28], on which the algorithm with fixed rates is based, the Powell method [47] was used as a local search method. However, in this work, the use of the Nelder Mead method [46] was preferred. The results for the Hybrid Genetic Algorithm with fixed rates (HGA-FR) and dynamic rates (HGA-DR) are presented in Table 5.5.

Table 5.5: Hybrid Genetic Algorithms Results with $N_{pop} = 100$

Function	Algorithm	f^*	k	Function Evaluation	Time	Euclidean Distance
Grlee	HGA-FR	-8.690×10^{-1}	1141	59339	2.23	1.303×10^{-5}
	HGA-DR	-8.690×10^{-1}	263	24306	0.84	1.380×10^{-5}
Forrester	HGA-FR	-6.021×10^{0}	851	44264	1.64	2.590×10^{-5}
	HGA-DR	-6.021×10^{0}	227	21297	0.73	2.688×10^{-5}
Branin	HGA-FR	3.979×10^{-1}	1040	54132	2.18	1.365×10^{-3}
	HGA-DR	3.979×10^{-1}	411	36439	1.36	1.163×10^{-3}
McCormick	HGA-FR	-1.913×10^{0}	897	46682	1.99	3.394×10^{-5}
	HGA-DR	-1.913×10^{0}	312	28219	1.10	3.656×10^{-5}
Easom	HGA-FR	-9.900×10^{-1}	1080	56191	2.29	2.601×10^{-2}
	HGA-DR	-9.900×10^{-1}	415	36789	1.39	2.601×10^{-2}
Ackley	HGA-FR	6.816×10^{-4}	2765	143848	5.88	2.930×10^{-4}
	HGA-DR	4.460×10^{-4}	881	75030	2.81	1.920×10^{-4}
Rastrigin	HGA-FR	9.270×10^{-5}	2605	135465	5.68	5.900×10^{-4}
	HGA-DR	1.788×10^{-4}	860	73268	2.80	7.990×10^{-4}
Rosenbrock	HGA-FR	1.512×10^{-9}	5404	281145	11.28	3.254×10^{-5}
	HGA-DR	1.594×10^{-9}	1936	161748	5.88	3.286×10^{-5}
S. Squares	HGA-FR	3.157×10^{-6}	1526	79397	3.31	9.340×10^{-4}
	HGA-DR	2.941×10^{-6}	803	68618	2.60	8.430×10^{-4}
Zakhavov	HGA-FR	1.080×10^{-6}	1829	95224	4.10	5.800×10^{-4}
	HGA-DR	8.578×10^{-7}	1032	87025	3.45	4.300×10^{-4}
Levy	HGA-FR	8.322×10^{-10}	872	45499	2.35	5.139×10^{-5}
	HGA-DR	7.581×10^{-10}	448	39389	1.87	4.878×10^{-5}
Schwefel	HGA-FR	6.364×10^{-5}	4101	213492	9.36	1.209×10^{-4}
	HGA-DR	6.364×10^{-5}	1872	156321	6.20	1.178×10^{-4}

The use of the hybrid method improved all function and algorithms results. The optimum solutions of the hybrid algorithms, Table 5.5, were better for all functions considered, in relation to the results of Table 5.3, where the hybrid approach was not used.

It is important to highlight the Rosenbrock, Zakahavov, Levy and Schwefel functions improvements in relation to the results presented in Table 5.3. In the Genetic Algorithm without local search method, GA-FR and GA-DR, the Rosenbrock function presented the optimum solutions the order of 10^{-1}, and in the hybrid strategy, HGA-FR and HGA-DR, the optimum solutions had the order of 10^{-9}. The improvement of Levy function was in the order of 10^{-6}, and the Zakhavov and Schwefel functions, were 10^{-3} and 10^{-2}, respectively.

A considered improvement was also noted in the Euclidean distance for all functions, meanly in those whose geometry is more complex. All improvements noted in Table 5.5 were done without compromise the time, the number of iterations and function evaluations, in relation to the results of Table 5.3.

Comparing only the HGA-FR and HGA-DR results, Table 5.5, on average the HGA-DR presented a reduced of 70% of the number of iterations and 40% less function evaluation, in relation to the HGA-FR.

Thereby, it is possible to conclude that the use of a local search method, after the GA performance, provides a considerable improvement in the algorithm solutions, without hard computational efforts. For this reason, the use of a local search method is recommended.

Chapter 6

Conclusion and Future Work

This work explored the Genetic Algorithm in order to propose new algorithms variations based on statistical measures that depend on the problem to solve. A new selection individual procedure was proposed based on the ability of each individual to generate offspring using conditional probability strategies. Besides, a new approach of the Genetic Algorithm with dynamic operators rates was presented, in which the rates are controlled by the population amplitude and the standard deviation. Furthermore, a new stopping criterion to be used in conjunction with the dynamic rates algorithm was presented. The validations of the proposed GA variants are done using twelve benchmark optimization functions.

Normally, the algorithms proposed in the literature are targeted to specific applications, since is a challenge to produce a robust algorithm to general use. Thus, the main difficulty found in this work was to propose a methodology that works for several functions, which simulated several real problems.

The selection approach using conditional probability did not present satisfactory results. Despite this strategy was able to select individuals, the results were not as expected since no improvement was noted in relation to the traditional approach.

The Genetic Algorithm with dynamic operators rates and the new stopping criterion presented excellent results. The algorithm proposed was able to determine the optimum solution with higher precision, in relation to the optimum solution presented in the literature. This approach used fewer iterations, functions evaluations and time than the Genetic Algorithm with fixed rates. When it was tested in a reduced population of 50 individuals, the dynamic rates algorithm also presented satisfactory results, however, the use of a 100 individuals is more indicated to avoid premature convergence when none specific information about the optimization problems is available.

When the algorithm with dynamic rates was compared with the default *Matlab* Genetic Algorithm implementation, it demonstrated superiority. The dynamic rates algorithm was able to find the optimum solutions for all functions considered. The default *Matlab* Genetic Algorithm implementation showed premature convergence and sometimes it got lost in the search space, without approaching the optimum solution.

The algorithm proposed was also tested with a local search method, in a hybrid approach, to improve the precision of the solution found by the Genetic Algorithm. This approach stands out in relation to the other approaches, since it improved the solution significantly, without so computational efforts.

Due to validation by functions of different properties, it can be concluded that the method of the dynamic algorithm is a competitive approach and it can be applied to several optimization problems.

As suggestion for future work can be considered, to explore the methodology proposed in this work to developing a new selection operator based on human genetic information transference.

Furthermore, it is possible to study machine learning strategies, as Fuzzy Systems and Artificial Neural Networks to identify patterns in a traditional version of the Genetic Algorithm and use the patterns to determine and control the possible alterations in each

operator rates.

Moreover, the Genetic Algorithm with dynamic rates can be applied to solve a real optimization problem and compare the results with a deterministic approach or other algorithms.

Besides, it is possible to apply the methodology of dynamic rates in Particular Swarm Optimization Algorithms.

Bibliography

[1] S. S. Rao, *Engineering optimization : theory and practice*, 4th ed. John Wiley & Sons, 2009, ISBN: 978-0-470-18352-6.

[2] M. Mitchell, *An Introduction to Genetic Algorithms*, 1st ed. Cambridge, MA, USA: MIT Press, 1998, ISBN: 0262631857.

[3] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, MA, USA: University of Michigan Press, 1992, ISBN: 0262082136.

[4] D. Pham and D. Karaboga, *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks*, 1st ed. Springer, 2000, ISBN: 978-1-447-11186-3. DOI: 10.1007/978-1-4471-0721-7.

[5] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, 1st ed. Springer, 2008, ISBN: 978-3-540-73189-4. DOI: 10.1007/978-3-540-73190-0.

[6] A. H. Gandomi, X. Yang, S. Talatahari, and A. H. Alavi, *Metaheuristic Applications in Structures and Infrastructures*, 1st ed. Elsevier, 2013, ISBN: 978-0-12-398364-0.

[7] J. Nocedal and S. J. Wright, *Numerical optimization*, 1st ed. Springer, 1999, ISBN: 0-387-98793-2.

[8] R. Čorić, M. Dumić, and D. Jakobović, "Complexity comparison of integer programming and genetic algorithms for resource constrained scheduling problems", IEEE, 2017 40th International Convention on Information, Communication Technology,

Electronics, and Microelectronics (MIPRO), 2017, pp. 1182–1188, ISBN: 978-953-233-090-8. DOI: 10.23919/MIPRO.2017.7973603.

[9] M. Giuzio, "Genetic algorithm versus classical methods in sparse index tracking", *Decisions in Economics and Finance*, vol. 40, pp. 243–256, 2017. DOI: 10.1007/s10203-017-0191-y.

[10] R. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, 2nd ed. John Wiley & Sons, 2004, ISBN: 0-471-45565-2.

[11] A. Ghaheri, S. Shoar, M. Naderan, and S. S. Hoseini, "The applications of genetic algorithms in medicine", *Oman Medical Journal*, vol. 30, no. 6, pp. 406–416, 2015. DOI: DOI10.5001/omj.2015.82.

[12] M. Fera, F. Fruggiero, A. Lambiase, R. Macchiaroli, and V. Todisco, "A modified genetic algorithm for time and cost optimization of an additive manufacturing single-machine scheduling", *International Journal of Industrial Engineering Computations*, vol. 9, no. 4, pp. 423–438, 2018. DOI: 10.5267/j.ijiec.2018.1.001.

[13] F. Alves, A. I. Pereira, A. Fernandes, and P. Leitão, "Optimization of home care visits schedule by genetic algorithm", in *Bioinspired Optimization Methods and Their Applications, Proceeding of 8th International Conference on*, Springer, 2018, pp. 1–12. DOI: 10.1007/978-3-319-91641-5_1.

[14] W. Li, Q. Zhou, J. Ren, and S. Samantha, "Data mining optimization model for financial management information system based on improved genetic algorithm", *Information Systems and e-Business Management*, pp. 1–19, 2019. DOI: 10.1007/s10257-018-00394-4.

[15] S. S. Patil and A. S. Bhalchandra, "Pattern recognition using genetic algorithm", IEEE, 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 310–314, ISBN: 978-1-5090-3243-3. DOI: 10.1109/I-SMAC.2017.8058361.

[16] J. Xu, L. Pei, and R. Zhu, "Application of a genetic algorithm with random crossover and dynamic mutation on the travelling salesman problem", *Procedia Comput. Sci.*, vol. 131, pp. 937–945, 2018, ISSN: 1877-0509. DOI: 10.1016/j.procs.2018.04.230.

[17] K. M., "The effects of a new selection operator on the performance of a genetic algorithm", *Applied Mathematics and Computation*, vol. 217, pp. 7669–7678, 2011. DOI: 10.1016/j.amc.2011.02.070.

[18] M. Akbari, H. Rashidi, and S. H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems", *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 35–46, 2017. DOI: 10.1016/j.engappai.2017.02.013.

[19] S. M. Lim, A. B. Sultan, N. S. Sulaiman, A. Mustapha, and K. Y. Leong, "Crossover and mutation operators of genetic algorithms", *International Journal of Machine Learning and Computing*, vol. 7, no. 1, pp. 9–12, 2017. DOI: 10.18178/ijmlc.2017.7.1.611.

[20] X. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, 1st ed. New Jersey, USA: John Wiley & Sons, Inc, 2010, ISBN: 9780470582466. DOI: 10.1002/9780470640425.

[21] A. Ritthipakdee, N. Thammano A. Premasathian, and B. Yyyanonvara, "A new selection operator to improve the performance of genetic algorithm for optimization problems", in *Proceedings of 2013 IEEE International Conference on Mechatronics and Automation*, IEEE, 2013, pp. 371–375. DOI: 10.1109/ICMA.2013.6617947.

[22] G. Pavai and T. V. Geetha, "A survey on crossover operators", *ACM Comput. Surv.*, vol. 49, no. 4, 72:1–72:43, 2016. DOI: 10.1145/3009966.

[23] L. Jánošíková, M. Herda, and M. Haviar, "Hybrid genetic algorithms with selective crossover for the capacitated p-median problem", *Central European Journal of Operations Research*, vol. 25, no. 1, pp. 651–664, 2017. DOI: 10.1007/s10100-017-0471-1.

[24] A. K. Das and D. K. Pratihar, "A directional crossover (dx) operator for real parameter optimization using genetic algorithm", *Applied Intelligence*, vol. 49, pp. 1841–1865, 2019. DOI: 10.1007/s10489-018-1364-2.

[25] M. Vannucci and V. Colla, "Fuzzy adaptation of crossover and mutation rates in genetic algorithms based on population performance", *Journal of Intelligent & Fuzzy Systems*, vol. 28, no. 4, pp. 1805–1818, 2015. DOI: 10.3233/IFS-141467.

[26] M. Kubichi and D. Figurowski, "An introduction to a novel crossover operator for real-value encoded genetic algorithm: Gaussian crossover operator", in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, IEEE, 2018, pp. 85–90. DOI: 10.1109/IIPHDW.2018.8388331.

[27] R. D. Priya and R. Sivaraj, "Dynamic genetic algorithm-based feature selection and incomplete value imputation for microarray classification", *Current Science*, vol. 112, no. 1, pp. 126–131, 2017. DOI: 10.18520/cs/v112/i01/126-131.

[28] D. Bento, D. Pinho, A. I. Pereira, and R. Lima, "Genetic algorithm and particle swarm optimization combined with powell method", in *11th International Conference of Numerical Analysis and Applied Mathematics*, vol. 1558, ICNAAM 2013, 2013, pp. 578–581. DOI: 10.1063/1.4825557.

[29] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer, 2005, ISBN: 0387212396.

[30] W. Lin, W. Lee, and T. Hong, "Adapting crossover and mutation rates in genetic algorithms.", *Journal of Information Science and Engineering*, vol. 19, no. 5, pp. 889–903, 2003.

[31] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems", PhD thesis, Universtity of Michigan, Ann Arbor, Michigan, 1975.

[32] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization", in *Proceedings of the Third International Conference on Genetic Algorithms*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 51–60.

[33] J. Grefenstette, "Optimization of control parameters for genetic algorithms", *IEEE Trans. Syst. Man Cybern.*, vol. 16, no. 1, pp. 122–128, 1986, ISSN: 0018-9472. DOI: 10.1109/TSMC.1986.289288.

[34] Q. Li, X. Tong, S. Xie, and G. Liu, "An improved adaptive algorithm for controlling the probabilities of crossover and mutation based on a fuzzy control strategy", in *Sixth International Conference on Hybrid Intelligent Systems (HIS'06)*, Rio de Janeiro, Brazil: IEEE, 2006. DOI: 10.1109/HIS.2006.264933.

[35] D. Whitley and T. Hanson, "Optimizing neural networks using faster, more accurate genetic search", in *Proceedings of the Third International Conference on Genetic Algorithms*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 391–396, ISBN: 1-55860-006-3.

[36] T. C. Fogarty, "Varying the probability of mutation in the genetic algorithm", in *Proceedings of the Third International Conference on Genetic Algorithms*, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 104–109, ISBN: 1-55860-006-3.

[37] T. Bäck and M. Schütz, "Intelligent mutation rate control in canonical genetic algorithms", in *Proceedings of the 9th International Symposium on Foundations of Intelligent Systems*, ser. ISMIS '96, London, UK, UK: Springer-Verlag, 1996, pp. 158–167, ISBN: 3-540-61286-6.

[38] H. Shimodaira, "A new genetic algorithm using large mutation rates and population-elitist selection (galme)", in *Proceedings of the 8th International Conference on Tools with Artificial Intelligence*, ser. ICTAI '96, Washington, DC, USA: IEEE Computer Society, 1996, pp. 25–32, ISBN: 0-8186-7686-8.

[39] M. Jamil and X. Yang, "A literature survey of benchmark functions for global optimization problems", *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013. DOI: 10.1504/IJMMNO.2013.055204.

[40] S. M. H. Mousavi, "Testing optimization algorithms easier with a new test function for single-optimization problems and validating evolutionary algorithms", *International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)*, vol. 8, no. 30, pp. 4009–4017, 2018.

[41] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster", *Transactions on Evolutionary computation*, vol. 3, no. 2, pp. 82–102, 1999. DOI: 10.1109/4235.771163.

[42] R. B. Gramacy and H. K. Lee, "Cases for the nugget in modeling computer experiments", *Statistics and Computing*, vol. 22, no. 3, pp. 713–722, 2012. DOI: 10.1007/s11222-010-9224-x.

[43] A. I. J. Forrester, A. Sóbester, and A. J. Keane, *Engineering Design via Surrogate Modelling A Practical Guide*, 1st ed. John Wiley & Sons, 2008, ISBN: N 978-0-470-06068-1.

[44] M. Laguna and R. Martí, "Experimental testing of advanced scatter search designs for global optimization of multimodal functions", *Journal of Global Optimization*, vol. 33, no. 2, pp. 235–255, 2005. DOI: 10.1007/s10898-004-1936-z.

[45] I. MathWorks, *Genetic algorithm - ga*, www.mathworks.com/help/gads/ga.html, Accessed: 2019-10-01.

[46] J. A. Nelder and R. Mead, "A simplex method for function minimization", *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965. DOI: doi.org/10.1093/comjnl/7.4.308.

[47] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives", *The Computer Journal*, vol. 7, no. 2, pp. 155–162, 1964. DOI: 10.1093/comjnl/7.2.155.

[48] S. Surjanovic and D. Bingham, *Virtual library of simulation experiments: Test function and datasets*, https://www.sfu.ca/ ssurjano/optimization.html, Accessed: 2019-03-01, Simon Fraser University.

Appendix A

Offspring Quality Matrix - OQM

To exemplify the (OQM) working, an initial population of 10 individuals and the Branin function are considered. Figure A.1 presents the initial OQM, which is composed by 10 individual ordered by their fitness in ascending order. Each matrix columns represents one individual and each row follow the description: $R1$ is the value of objective function; $R2$ is the number of times that the individual was used in the crossover procedure; $R3$ is the number of high quality offspring generated in crossover, that is, the number of times that the offspring generated had objective function value smaller than the C_f, in the crossover procedure; $R4$ is the number of the times that the individual was used in the mutation procedure; $R5$ is the number of high quality offspring in the mutation procedure. The same description of rows and columns is valid for Figures A.2 and A.3.

Considering natural selection rate equal $0.5 \times N_{pop}$, it results in $N_{keep} = 5$, so the 5 best individuals of the population are selected to be possible parents of this generation. In this example the top 5 individuals are in gray in Figure A.1 and the C_f is equal to 54.5350, presented by individual of $C5$ column.

The crossover rate utilized is equal to $0.25 \times N_{pop}$, so the crossover procedure will be responsible for generating 3 new offspring. If the number of individuals selected to be

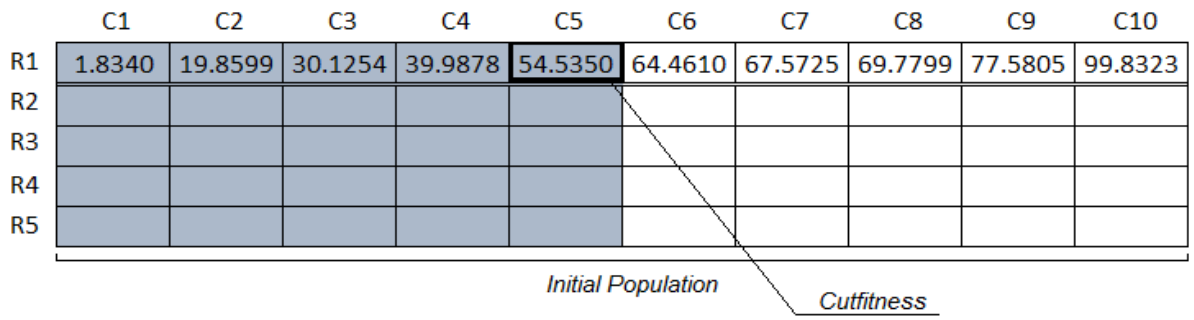

Figure A.1: Initial Population

parents is odd, another individual is chosen inside the subset N_{keep} to be a parent and complete the pair. Note, in this case, one individual can be selected more than one time in the same generation. After an offspring be created in the crossover procedure, the OQM row $R2$ of the parents, sums 1, for each offspring generated. Then, the value of its fitness is compared with the current C_f. When an offspring has the fitness smaller than C_f the parents OQM row $R3$ sums 1. In this example, as shown in Figure A.2 the individuals of the columns $C1$, $C2$ and $C5$ were used as parents and the individual of columns $C6$, $C7$, $C8$ are the crossover offspring. When compared the offspring fitness with the current C_f, it is possible to note that the individual of $C8$ column, has fitness smaller than the C_f value, for this reason, its parents, which is represented in the columns $C2$ and $C5$ sums 1 in their OQM row $R3$. The three new individual offspring generated in the crossover are shown in blue in Figure A.2.

The mutation rate is also $0.25 \times N_{pop}$, consequently the other three new offspring are generated by mutation. Similar to the crossover process, for mutation, when an individual is chosen to be muted, 1 is summed in its the mutation OQM row $R4$. And, when the offspring have fitness better than the C_f, 1 is summed in the parents OQM row $R5$, which represents the number of high quality offspring generated by the mutation. As we can seen in Figure A.2, the individual represented in the columns $C2$, $C3$ and $C5$ are chosen to be muted and they originated the offspring represented in the columns $C9$, $C10$ and $C11$, which are in yellow in Figure A.2. Among these three offspring individuals only the

individual in the column $C10$ has fitness smaller than the C_f, thus its parent $C2$ sums 1 in $R5$, as presented in Figure A.2.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
R1	1.8340	19.8599	30.1254	39.9878	54.5350	67.275	144.6816	27.3878	55.457	18.6622	106.7405
R2	1	3	0	0	2	0	0	0	0	0	0
R3	0	1	0	0	1	0	0	0	0	0	0
R4	0	1	1	0	1	0	0	0	0	0	0
R5	0	1	0	0	0	0	0	0	0	0	0

Candidate Parents *Crossover Offspring* *Mutation Offspring*

Figure A.2: Intermediate Population

Thereafter, the offspring generated in the crossover and mutation procedures are added in the current population N_{pop}. Considering the used rates and consequently the number of individual in the initial population, at the end of the first iteration will have 16 individuals: 10 from the initial population 3 from crossover and 3 from the mutation procedure. These individuals order according to their fitness, as Figure A.3. The top 10 are considered the new generation N_{pop}, of the next algorithm interaction, while the last 6 are removed. These processes are repeated until a stopping criterion is achieved.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
R1	1.8340	18.6622	19.8599	27.3878	30.1254	39.9878	54.5350	55.457	64.4610	67.275	67.5725	69.7799	77.5805	99.8323	106.7405	144.6816
R2	1	0	3	0	0	0	2	0		0					0	0
R3	0	0	1	0	0	0	1	0		0					0	0
R4	0	0	1	0	1	0	1	0		0					0	0
R5	0	0	1	0	0	0	0	0		0					0	0

Next Generation *Individuals Removed*

Figure A.3: Final Population of the First Iteration

As said before, the Figures A.1, A.2 and A.3 are just to exemplify the idea used in this work. The real population considered is constituted by 100 individuals, and it was not

presented here due to the OQM dimensions.

Appendix B

Benchmark Optimization Functions

Gramacy and Lee Function - Grlee

Multimodal, continuous, differentiable.

$$f(x) = \frac{sin(10\pi x)}{2x} + (x-1)^4. \tag{B.1}$$

Number of variables $N_{var} = 1$.

Input domain: $x \in [0.5, 2.5]$.

Global minimum: $f(x^*) = -0.869011$, at $x^* = (0.548563)$.

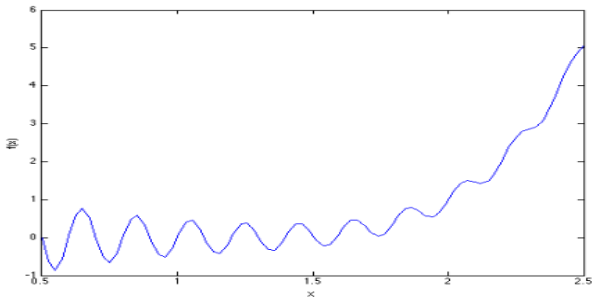

Figure B.1: Gramacy and Lee Function with $N_{var} = 1$
[48]

Forrester Function

Multimodal, continuous, differentiable.

$$f(x) = (6x - 2)^2 sin(12x - 4). \tag{B.2}$$

Number of variables $N_{var} = 1$.

Input domain: $x \in [0, 1]$.

Global minimum: $f(x^*) = -6.020707$, at $x^* = 0.757000$

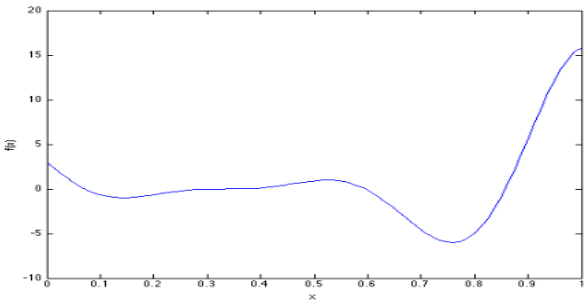

Figure B.2: Forrester Function with $N_{var} = 1$
[48]

Branin Function

Multimodal, continuous, differentiable, non-separable, non-scalable and non-convex

$$f(x) = 1(x_2 + \frac{5.1}{4\pi^2}x_1^2 - \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x_1) + 10. \tag{B.3}$$

Number of variables $N_{var} = 2$.

Input domain: $x_1 \in$ [-5, 10], $x_2 \in$ [0, 15].

Global minimum: $f(x^*) = 0.397887$, at $x^* = (-\pi, 12.275), (\pi, 2.2775)$ and $(9.42478, 2.475)$.

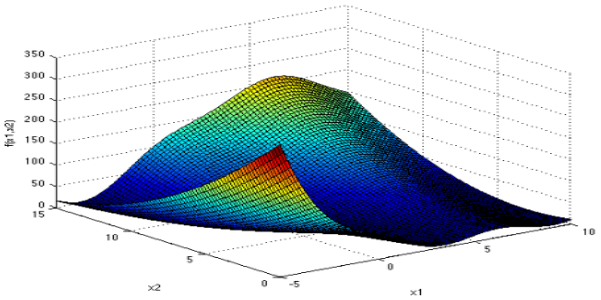

Figure B.3: Branin Function with $N_{var} = 2$
[48]

McCormick Function

Multimodal, continuous, differentiable, non-separable, non-scalable and non-convex

$$f(x) = sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1. \tag{B.4}$$

Number of variables $N_{var} = 2$.

Input domain: $x_1 \in$ [-1.5, 4], $x_2 \in$ [-3, 4].

Global minimum: $f(x^*) = -1.9133$, at $x^* = (-0.54719, -1.54719)$.

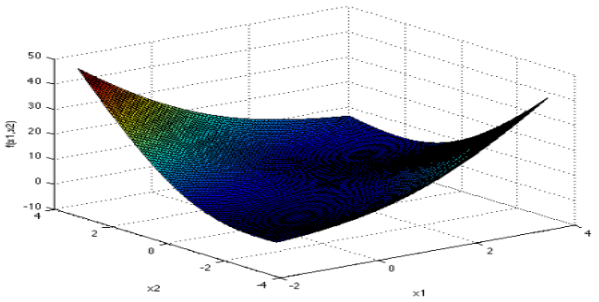

Figure B.4: McCormick Function with $N_{var} = 2$
[48]

Easom Function

Multimodal, continuous, differentiable, non-separable, non-scalable and non-convex

$$f(x) = -cos(x_1)cos(x_2)exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2) \tag{B.5}$$

Number of variables $N_{var} = 2$.

Input domain: $x_i \in$ [-10, 10], for all i=1, 2.

Global minimum: $f(x^*) = -1$, at $x^* = (\pi, \pi)$.

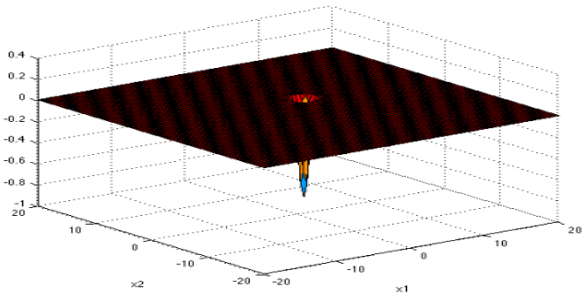

Figure B.5: Easom Function with $N_{var} = 2$
[48]

Ackley Function

Multimodal, continuous, differentiable, non-separable, scalable and non-convex

$$f(x) = -20exp\left(-0.2\sqrt{\frac{1}{N_{var}}\sum_{i=1}^{N_{var}} x_i^2}\right) - exp\left(\frac{1}{N_{var}}\sum_{i=1}^{N_{var}} cos(2\pi x_i)\right) + 20 + exp(1). \quad \text{(B.6)}$$

Number of variables $N_{var} = 3$.

Input domain: $x_i \in$ [-32.768, 32.768], for all i=1, ..., N_{var}.

Global minimum: $f(x^*) = 0$, at $x^* = (0, ..., 0)$.

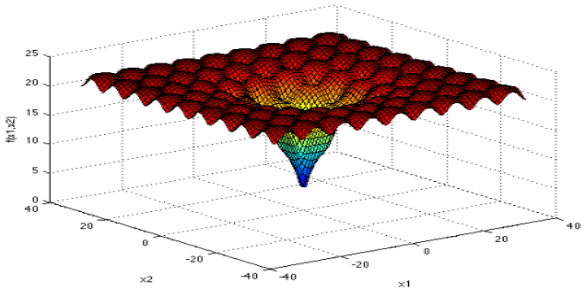

Figure B.6: Ackley Function with $N_{var} = 2$
[48]

Rastrigin Function

Multimodal, continuous, differentiable, separable, scalable and convex

$$f(x) = 10N_{var} + \sum_{i=1}^{N_{var}} [x_i^2 - 10cos(2\pi x_i)]. \tag{B.7}$$

Number of variables $N_{var} = 3$.

Input domain: $x_i \in$ [-5.12, 5.12], for all i=1, ..., N_{var}.

Global minimum: $f(x^*) = 0$, at $x^* = (0, ..., 0)$.

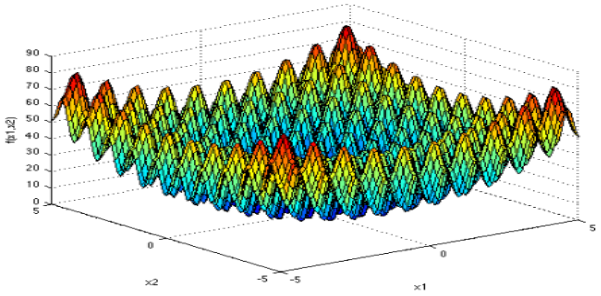

Figure B.7: Rastrigin Function with $N_{var} = 2$
[48]

Rosenbrock Function

Unimodal, continuous, differentiable, non-separable, scalable and non-convex.

$$f(x) = \sum_{i=1}^{N_{var}-1} [100(x_{i+1}) - x_i^2)^2 + (x_i - 1)^2] \tag{B.8}$$

Number of variables $N_{var} = 3$.

Input domain: $x_i \in$ [-5, 10], for all i=1, ..., N_{var}.

Global minimum: $f(x^*) = 0$, at $x^* = (1, ..., 1)$.

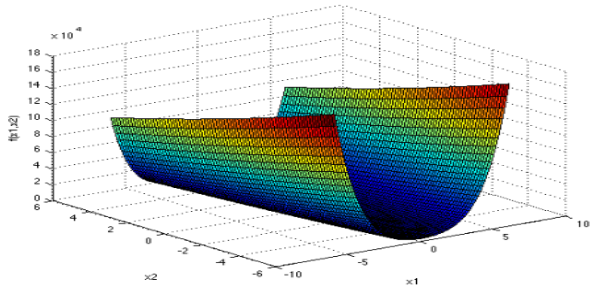

Figure B.8: Rosenbrock Function with $N_{var} = 2$
[48]

Sum Squares Function

Unimodal, continuous, differentiable, separable, scalable and convex

$$f(x) = \sum_{i=1}^{N_{var}} i x_i^2. \tag{B.9}$$

Number of variables $N_{var} = 4$.

Input domain: $x_i \in$ [-10, 10], for all i=1, ..., N_{var}.

Global minimum: $f(x^*) = 0$, at $x^* = (0, ..., 0)$.

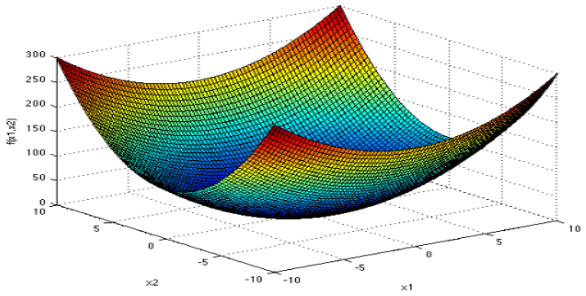

Figure B.9: Sum Squares Function with $N_{var} = 2$
[48]

Zakharov Function

Multimodal, continuous, differentiable, separable, scalable and convex

$$f(x) = \sum_{i=1}^{N_{var}} x_i^2 + \left(\sum_{i=1}^{N_{var}} 0.5 i x_i \right)^2 + \left(\sum_{i=1}^{N_{var}} 0.5 i x_i \right)^4 . \tag{B.10}$$

Number of variables $N_{var} = 4$.

Input domain: $x_i \in$ [-5, 10], for all i=1, ..., N_{var}.

Global minimum: $f(x^*) = 0$, at $x^* = (0, ..., 0)$.

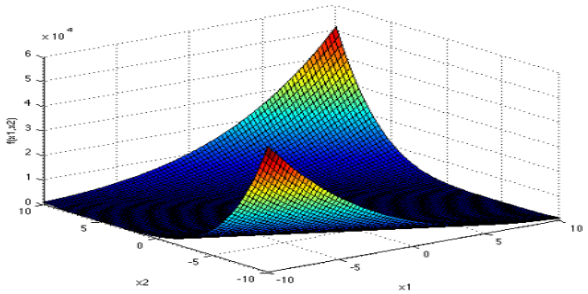

Figure B.10: Zakharov Function with $N_{var} = 2$
[48]

Levy Function

Multimodal, continuous, differentiable, separable and non-convex

$$f(x) = sin^2(\pi w_1) + \sum_{i=1}^{N_{var}-1} (w_i - 1)^2[1 + 10sin^2(\pi w_i + 1)] + (w_{N_{var}} - 1)^2[1 + sin^2(2\pi w_{N_{var}})], \tag{B.11}$$

where $w_i = 1 + \frac{x_i - 1}{4}$, for all i = 1, ..., N_{var}.

Number of variables $N_{var} = 5$.

Input domain: $x_i \in$ [-10, 10], for all i=1, ..., N_{var}.

Global minimum: $f(x^*) = 0$, at $x^* = (1, ..., 1)$.

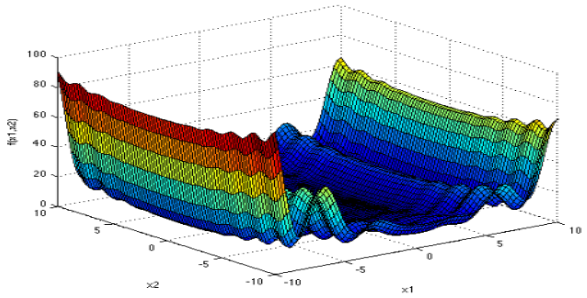

Figure B.11: Levy Function with $N_{var} = 2$
[48]

Schwefel Function

Multimodal, continuous, differentiable, separable, scalable and convex

$$f(x) = 418.9829N_{var} - \sum_{i=1}^{N_{var}} x_i sin(\sqrt{|x_i|}). \tag{B.12}$$

Number of variables $N_{var} = 5$.

Input domain: $x_i \in$ [-500, 500], for all i=1, ..., N_{var}.

Global minimum: $f(x^*) = 0$, at $x^* = (420.9687, ..., 420.9687)$.

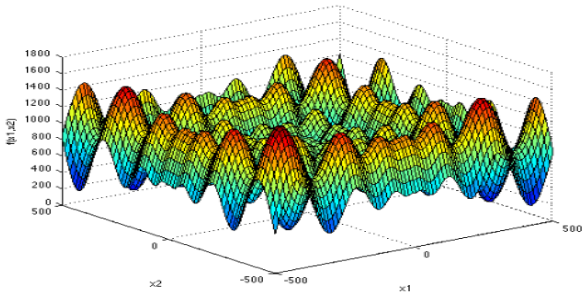

Figure B.12: Schwefel Function with $N_{var} = 2$
[48]