UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LEONARDO DE LIMA CORRÊA

A Memetic Algorithm Framework for Multimodal Continuous Optimization

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Advisor: Prof. Dr. Márcio Dorn

Porto Alegre
July 2022

CIP — CATALOGING-IN-PUBLICATION

Corrêa, Leonardo de Lima

A Memetic Algorithm Framework for Multimodal Continuous Optimization / Leonardo de Lima Corrêa. – Porto Alegre: PPGC da UFRGS, 2022.

255 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2022. Advisor: Márcio Dorn.

1. Multimodal optimization. 2. Metaheuristic. 3. Evolutionary algorithm. 4. Swarm intelligence. 5. Knowledge-based memetic algorithm. 6. Structural bioinformatics. I. Dorn, Márcio. II. Título.

ABSTRACT

Despite the advances in computational methods and the wide range of metaheuristics proposed for multimodal continuous optimization, there is still a strong demand for developing new strategies focused on issues related to the algorithms' performance when applied to challenging problems with complicated objective functions. The general idea of this work is centered around the investigation of distinct metaheuristic characteristics to deal with multimodal problems in the continuous domain. In this sense, we defined the multimodal 3-D protein structure prediction problem as our real case study, one of the most important problems in Structural Bioinformatics. Thus, we proposed an adaptive memetic algorithm as a general framework-based method with multiple populations and niching strategies, which incorporates concepts of bio-inspired algorithms for global optimization with separate local improvement. To evaluate the proposed approach, we designed different versions of the framework for three scenarios of multimodal optimization: (*i*) the general framework for single global continuous optimization with multimodal objective function; (*ii*) the framework with archive strategy for multimodal optimization with more than one global optimum; and (*iii*) the framework with specific-problem components for the multimodal problem of predicting the 3-D protein structures. With the development of this work, we aimed to create, via a constructive and incremental approach, an evolutionary method capable of dealing with the inherent multimodality and issues of a range of optimization functions while preserving accurate results. Our focus was also to evaluate the behavior of the presented methods and search components facing multiple multimodal problems. The memetic algorithm framework was able to perform well on all the optimization scenarios explored by reaching promising results compared with relevant methods related to the corresponding research fields. Nonetheless, despite the obtained results, we highlight that each implemented algorithmic version still needs improvements regarding each of the delineated case studies to further enhance the method's performance and results.

Keywords: Multimodal optimization. metaheuristic. evolutionary algorithm. swarm intelligence. knowledge-based memetic algorithm. structural bioinformatics.

Proposta de um *Framework* Baseado em Algoritmo Memético para Otimização de Problemas Multimodais

RESUMO

Apesar dos avanços computacionais e da ampla gama de meta-heurísticas propostas na literatura para lidar com problemas de otimização multimodal, ainda existe a necessidade de desenvolver novas estratégias de busca voltadas ao desempenho dos algoritmos quando aplicados a problemas difíceis com complexas funções de avaliação. A ideia geral deste trabalho consiste na investigação de diferentes aspectos relativos as meta-heurísticas utilizadas para tratar problemas multimodais de domínio contínuo. Neste sentido, definiu-se, como estudo de caso principal para o trabalho, a predição de estruturas 3-D de proteínas, o qual representa um dos mais importantes problemas da Bioinformática Estrutural. Neste trabalho, foi proposto um algoritmo memético adaptativo na forma de um *framework* computacional de propósito geral, o qual incorpora múltiplas populações, conceitos de algoritmos evolutivos e inteligência de enxame, combinados a funções adicionais de busca local, controle de convergência e performance. Como forma de avaliação do método proposto, foram idealizadas diferentes versões, as quais foram empregadas em três cenários distintos de otimização: (*i*) versão mais geral para tratar funções multimodais com um único ótimo global; (*ii*) versão com arquivamento externo de soluções focada em otimização multimodal com diversos ótimos globais; e (*iii*) versão baseada em conhecimento com componentes de busca específicos para lidar com o problema de predição de estruturas 3-D de proteínas. Com isso, objetivou-se prover, através do desenvolvimento de uma abordagem incremental, um método de busca para lidar com as complexidades relativas à multimodalidade inerente a diversos problemas de otimização. De maneira geral, concluiu-se que a abordagem proposta obteve êxito quanto à otimização dos problemas concernentes aos diferentes cenários de otimização idealizados. O método atingiu resultados promissores em relação aos algoritmos mais relevantes das áreas relativas a cada estudo de caso. No entanto, ressalta-se que cada uma das versões implementadas ainda necessita de melhorias em relação a cada um dos estudos de caso delineados, objetivando aprimorar ainda mais o desempenho obtido, bem como os resultados do método como um todo.

Palavras-chave: Otimização multimodal, algoritmos evolutivos, inteligência de enxame, algoritmo memético baseado em conhecimento, bioinformática estrutural.

LIST OF ABBREVIATIONS AND ACRONYMS

3-D	Three-Dimensional
NMR	Nuclear Magnetic Resonance
EM	Electron Microscopy
PSP	Protein Structure Prediction
FM	Free-Modeling
RefSeq	NCBI Reference Sequence Database
PDB	Protein Data Bank
NFL	No Free Lunch
EA	Evolutionary Algorithm
SI	Swarm Intelligence
MA	Memetic Algorithm
LS	Local Search
ABC	Artificial Bee Colony
SW	Solis and Wets
RTS	Restricted Tournament Selection
Expr/Expt	Exploration and Exploitation
CV	Coefficient of Variation
SCN	Similarity to the Closest Neighbor
PS	Primary Structure
SS	Secondary Structure
TS	Tertiary Structure
QS	Quaternary Structure
IDP	Intrinsically Disordered Proteins
SCOP	Structural Classification of Protein Database

RG	Radius of Gyration
CM	Protein Contact Map
CASP	Critical Assessment of Protein Structure Prediction
SA	Simulated Annealing
AI	Artificial Intelligence
ES	Evolution Strategies
GA	Genetic Algorithm
DE	Differential Evolution
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
GABC	Gbest-Guided ABC
IABC	Improved ABC
ILABC	Information Learning ABC
DFS	Depth-First Search
GRO	Gene Recombination Operator
BFGS	Broyden-Fletcher-Goldfarb-Shanno
AMPS	Adaptive Method for the Population Size
SABC-GB	Self-Adaptive ABC based on the Gbest
CSGS	Candidate Solution Generating Strategy
DNABC	Dynamic Best Neighbor-Guided ABC
NABC	Best Neighbor-Guided ABC
HABCDE	Hybrid ABC with DE
sdABC	Self-Adaptive Differential ABC
CEC	Congress on Evolutionary Computation
SHADE	Success-History based Adaptive DE

LPSR	Linear Population Size Reduction Mechanism
GECCO	Genetic and Evolutionary Computation Conference
SEMCCO	Swarm, Evolutionary and Memetic Computing Conference
CF	Crowding Factor
DC	Deterministic Crowding
HV	Hill-Valley
LIPS	Locally Informed Particle Swarm
CDE	Crowding DE
SDE	Species-based DE
NCDE	Neighborhood based CDE
NSDE	Neighborhood based SDE
NShDE	Neighborhood based Sharing DE
DSDE	Dual-Strategy DE
APC	Affinity Propagation Clustering
AM-ACO	Adaptive Multimodal ACO
RS-CMSA	Covariance Matrix Self-adaption Evolution Strategy with Repelling Sub-populations
EDA	Estimation of Distribution Algorithm
MEDA	Multimodal EDA
HillVallEA	Hill-Valley EA
ANDE	Automatic Niching DE
TLLS	Two-Level Local Search
CPA	Contour Prediction Approach
MaHDE	Multi-Angle Hierarchical Differential Evolution
FHM	Fitness Hierarchical Mutation
DGS	Directed Global Search

ELS	Elite Local Search
DIDE	Distributed Individuals Differential Evolution
DIMP	Distributed Individuals for Multiple Peaks
ELM	Elite Learning Mechanism
AGDE	Adaptive Guidance-based Differential Evolution
AMS	Adaptive Mutation Strategy
IFA	Iterative Feedback Archive
GDEL	Gaussian Disturbance-based Elite Learning
SDDLCSDE	Self-Adaptive Double-Layer-Clustering Speciation Differential Evolution
UCT	Upper Confidence Tree
MCTS	Monte Carlo Tree Search
RDS	Random Restarts with Decreasing Step-Size
NBC	Nearest-Better Clustering
WGraD	Weighted Gradient and Distance-based Clustering method
MST	Minimum Spanning Tree
MSTDE	MST with DE
DPR	Dynamic Pruning Ratio
VM	Virtual Machine
MMDE	Memetic Differential Evolution
XLS	Crossover-based Local Search
RMAwA	Region-based Memetic Algorithm with Archive
REMC	Replica Exchange Monte Carlo
MSA	Multiple Sequence Alignment
AMW	Associative Memory Hamiltonian with Water
APL	Angle Probability List
NSGA-II	Non-Dominated Sorting Genetic Algorithm

LIP	Local Improvement Procedure
DNSP	Dynamic Niche Size Procedure
TRP	Tree Resizing Procedure
SMA	Simple Moving Average
RMSD	Root-Mean-Square-Deviation
PR	Peak Ratio
SR	Success Rate
GDT_TS	Global Distance Total Score test

LIST OF SYMBOLS

ϕ	Angle Phi
ψ	Angle Psi
ω	Angle Omega
χ	Angle Chi

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

General-purpose optimization is the effort of seeking the best possible solution among multiple alternatives to a given problem. In many cases, the dimensions of theoretical and commercial optimization problems may grow too fast, characterizing a hard optimization issue (TEZEL; MERT, 2020). Hard optimization, which may be continuous optimization, is concerned with problems that cannot be optimally solved, or to any guaranteed bound, by any exact method within a reasonable time limit (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). Depending on the complexity of the problem under study, it can be solved through deterministic methods or heuristic-based solutions. On one hand, exact algorithms can achieve optimal results. However, when applied to hard problems (COOK, 1983), they present non-polynomial execution time, which is computationally impracticable. On the other hand, heuristic or approximate methods, if well-designed, can reach sufficiently good solutions within a reasonable execution time when applied to real-life problems. However, they do not guarantee optimal results (TALBI, 2009).

Real-life problems often represent challenging problems. Thus, researchers have been proposed several methods to deal with these issues, and among them, stochastic metaheuristics are being widely explored (DRÉO et al., 2006; BOUSSAÏD; LEPAGNOT; SIARRY, 2013; LUKE, 2013). Metaheuristics are practically problem-independent techniques and are defined as high-level heuristics that can be used in a wide range of domains with minor modifications in their components. Furthermore, such algorithms have been successfully applied into a variety of real-life domains in academia and industry, in areas ranging from finance to production management, life sciences, and engineering (BOUSSAÏD; LEPAGNOT; SIARRY, 2013; SER et al., 2019; HUSSAIN et al., 2019).

1.1 Research Problem

Among the several application of metaheuristics, one of them encompasses the research field of Structural Bioinformatics, which corresponds to the study area focused on the three-dimensional (3-D) structure of molecules and macromolecules (CHOU, 2004), including 3-D protein structure prediction (DILL; MACCALLUM, 2012), molecular docking and modeling (YURIEV; HOLIEN; RAMSLAND, 2015), and studies about the relationship between protein structure and its function (WHISSTOCK; LESK, 2003). The structural information corresponding to each molecule, such as DNA, protein, ligand, are mainly obtained through experimental methods, such as X-ray crystallography (MCREE, 1999), Nuclear Magnetic Resonance (NMR) (CAVANAGH et al., 2006) and Electron Microscopy (EM) (UNWIN; HENDERSON, 1975).

Currently, regarding the research scenarios of Structural Bioinformatics, many problems remain partially unsolved, such as the molecular docking (YURIEV; HOLIEN; RAMSLAND, 2015) and the 3-D protein structure prediction (PSP) (DILL; MACCALLUM, 2012; DORN et al., 2014). Some of the reasons in which these problems impose significant challenges are due to the high cost and considerable required time of experimental methods, high computational complexity, and also the lack of complete comprehension of the rules that conduct the biochemical processes and their relations (ANFINSEN, 1973). Specifically, PSP configures a key issue in this field and describes the efforts to develop computational strategies to determine the 3-D protein structure (DILL; MACCALLUM, 2012). A single sequence of chained amino acids defines a protein that under specific physiological conditions folds into a particular conformation (ANFINSEN, 1973). Proteins are in all living systems and perform an extensive set of fundamental life functions, where the protein function's nature is directly related to its assumed 3-D structure. Then, protein folding can provide valuable understandings about the protein roles in the cell (BRANDEN; TOOZE, 1999; LASKOWSKI; WATSON; THORNTON, 2005).

Over the last decades, the PSP focused on the structure modeling just from the amino acid (aa) sequence (free-modeling (FM) category) has challenged biologists, biochemists, physicists, computer scientists, and mathematicians, remaining a critical challenge in Structural Bioinformatics (BAXEVANIS; OUELLETTE, 2004). The problem is classified according to the computational complexity theory (COOK, 1983) as NP-hard problem (UNGER; MOULT, 1993; CRESCENZI et al., 1998) due to the high dimensionality of variables and its multimodal search space, presenting an exponential growth of difficulty as the amino acid number increases (GUYEUX et al., 2014). The problem complexity relies on the combinatorial explosion of plausible conformations, where an amino acid chain can give rise to a few structures around native states among several existing possibilities (BAXEVANIS; OUELLETTE, 2004).

As consequence of the difficulty in predicting the 3-D protein structures by experimental methods, there is a considerable gap between the data volume (non-redundant aa sequences) generated through Genome Projects, which are stored in the NCBI Reference Sequence Database[1] (*RefSeq*) (PRUITT; TATUSOVA; MAGLOTT, 2005) and the exper-

[1]<www.ncbi.nlm.nih.gov/refseq>

imentally determined and non-redundant 3-D structures stored in the Protein Data Bank[2] (PDB) (BERMAN et al., 2000). Currently, less than 1% of known and non-redundant protein sequences have representatives in the PDB (BERMAN et al., 2000). RefSeq and PDB are currently the largest and most popular databases for storing aa sequences, DNA genomics, non-redundant transcripts, and 3-D protein structures, respectively.

The modeling process of protein structures as computational optimization can be seen as a way to overcome some of the PSP complexities and ease the protein structure-based studies. Therefore, several methods have been proposed to address the problem (DORN et al., 2014). These methods can be classified, but not strictly, into two different classes, where they are grouped concerning the use or not of structural information from the PDB: (*i*) first principle methods or *ab initio* (OSGUTHORPE, 2000); and (*ii*) fold recognition and comparative modeling methods (BOWIE; LUTHY; EISENBERG, 1991; MARTÍ-RENOM et al., 2000). Specifically, in this work, we are interested in a group of methods located between these two classes, which consists of a hybrid class of knowledge-based methods that make use of template information from experimental protein structures associated with an *ab initio* strategy based on simulations of physicochemical properties of the folding process in nature (ROHL et al., 2004). Thus, to predict 3-D protein structures adopting these concepts, a wide range of stochastic optimization algorithms and metaheuristics are being proposed to find approximate solutions to the PSP (DORN et al., 2014). As already introduced, such techniques do not always guarantee the optimal solution, but they provide a reasonable approximation with a limited computational effort (TALBI, 2009). Also, the knowledge incorporation of protein structures from the PDB represents a critical strategy to support the modeling methods, reducing the conformational search space size (ABRIATA et al., 2018; MOULT et al., 2018; KRYSHTAFOVYCH et al., 2019; KUHLMAN; BRADLEY, 2019).

1.2 Research Motivation

Regardless of the computational advances to deal with the PSP, it lasts an open challenge. The development of novel approaches from biological data sources combined with state-of-the-art computational techniques, such as optimization and machine learning methods, have shown relevant results in recent years and should be further explored in the coming years so that computer simulation is increasingly closer to reality (BRADLEY;

[2]<www.rcsb.org>

MISURA; BAKER, 2005; DORN et al., 2014).

It is well known that the PSP success depends on an accurate energy function that may reflect the native state of conformations (KIM et al., 2009), as well as a robust metaheuristic in search space exploration and maintenance of solutions' diversity throughout the prediction process due to the evaluation function's multimodality and high dimensionality of variables (GARZA-FABRE et al., 2016). Thus, combining these factors with knowledge-based strategies from previously known protein structures, we believe that through the development of search algorithms focused on multimodal optimization problems and their adaptation to the PSP issues, it is possible to enhance the method's potential while obtaining satisfactory outcomes for the problem (ISLAM; CHETTY, 2009).

Furthermore, especially in real-life problems as the PSP, the simple use of canonical metaheuristics does not always present the expected behavior. The main reasons are the severe roughness (multimodality) of the problem fitness landscape and the complexity of managing local and global points of the objective function, where, in this case, even a small chain of amino acids can assume several conformations (BELDA et al., 2007; HANDL; LOVELL; KNOWLES, 2008; GARZA-FABRE et al., 2016).

The success of metaheuristic design on a given optimization problem is defined primarily if the search process can provide an efficient search strategy and a satisfactory balance (trade-off) between search space exploration and exploitation. The exploration process is related to the solutions' diversification and aims to identify regions of the search space with high-quality solutions. In contrast, the exploitation of solutions is the search intensification around promising areas of the accumulated search experience (ČREPINŠEK; LIU; MERNIK, 2013; XU; ZHANG, 2014). The main differences between them are related to the particular way in which they try to reach such goals, wherein several algorithms are being proposed to achieve these needs. Metaheuristics are classified according to many criteria, but a fundamental distinction in the literature is the differentiation between single-solution and population-based algorithms. The most studied population-based methods are related to bio-inspired concepts of evolutionary algorithms (EAs) and swarm intelligence (SI) (LUKE, 2013; SER et al., 2019). In general, population-based algorithms are more exploration-oriented, whereas single-solution metaheuristics are more exploitation oriented (BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

It is well-known that metaheuristics as strategies of optimization are suitable for many real-world applications. Nevertheless, according to the No Free Lunch (NFL) theorem, proposed by Wolpert and Macready (WOLPERT; MACREADY, 1997), there is not any optimization algorithm able to present higher performance than others under any metric in solving all possible optimization problems (BEHESHTI; SHAMSUDDIN, 2013; SER et al., 2019). With this, it can be stated that there are optimization algorithms better than others for specific application domains. Still, according to the NFL theorem, to improve the average performance of a given method, it should use previous information and search components focused on the problem under study or on similar problems from the same domain to validate the proposed search method to the problem.

Existing metaheuristics suffer from some drawbacks such as slow convergence rate, trapping into local optima and weak diversity, complex search operators, need to control many parameters and adapt for specific search spaces (BEHESHTI; SHAMSUDDIN, 2013). Hence, in-depth investigations around metaheuristic design are needed to overcome such problems and minimize the disadvantages focused on specific optimization domains.

1.3 Research Scope

This thesis delimits the work's scope regarding the multimodal continuous optimization domain (DAS et al., 2011; LI et al., 2016). Essentially, multimodal optimization aims to overcome obstacles imposed by the functions' multimodality over adaptations in the algorithms. Its goal is to find a variety of optimal or suboptimal solutions and not just a single solution to the problem (DAS et al., 2011). However, according to literature, multimodal strategies are also used to tackle global continuous optimization problems, seeking to find a single global optimum solution on a multimodal search space (LI et al., 2016). Both scenarios of multimodal optimization are addressed in this work.

Within this context, we highlight the points of interest of this work related to the metaheuristic design: (*i*) discovery and maintenance of the search space optimal or suboptimal points through multimodal strategies (LI et al., 2016); (*ii*) the balance between exploration and exploitation search operations (ČREPINŠEK; LIU; MERNIK, 2013); and (*iii*) the parameter control problem (KARAFOTIAS; HOOGENDOORN; EIBEN, 2014; PARPINELLI et al., 2019). The first point concerns the adoption of multimodal metaheuristics and strategies to explore better and refine the search space to find and maintain the existing local and global optima over the search process. The second one is addressed in terms of the effectiveness of exploration and exploitation efforts. Depending on the defined trade-off, it can lead the optimization process to a high diversity and slow convergence rate, low diversity and algorithm stagnation, or unnecessary computation, among other scenarios. Regarding the third point, the parameter control problem deals with the careful variation of the parameter values over the run of a metaheuristic since the algorithm performance greatly depends on the values of its parameters. This point can be investigated through adaptive mechanisms, which allow the method to use satisfactory dynamic parameterization in different stages of the process.

However, it is noteworthy that one point is strictly related to the others (SER et al., 2019). For instance, if the first point fails, the effort to ensure an appropriate trade-off between exploration and exploitation, and parameterization becomes irrelevant. Alternatively, suppose the method does not present a satisfactory balance in the second point. In that case, a multimodal search strategy may be useless since the method can reach a premature convergence or never converge. Also, parameter selection is directly responsible for the method's performance. Depending on the parameterization and the search operators used, they can influence the optimization process, such as population diversity and convergence rates. These concerns represent some of the most critical challenges of the population-based algorithms, and addressing them means investigating a considerable part of the concerns when designing a metaheuristic. So the decision-making over one issue may cause an impact on all other method's components and, consequently, on the performance measures. Hence, the address of a single issue may determine the success of others, which is why they should be investigated together.

Moreover, it is also necessary to consider the NFL theorem as more algorithms are being proposed. These methods should be tested in a real-life problem, which can be distinct in terms of complexity from the benchmarks functions. The validation on benchmark problems may somehow show that a metaheuristic is robust enough. However, it is not guaranteed that the performance will remain the same when solving real problems (WONG; MING, 2019; TZANETOS; DOUNIAS, 2020).

1.4 Research Proposal

The general idea of this thesis is centered around the investigation of distinct metaheuristic's characteristics to deal with optimization problems in the multimodal continuous domain. To do so, we defined the NP-hard problem of predicting the 3-D protein structures as our real case study. Based on this, we intend to address the PSP from a constructive perspective, starting from more general benchmark optimization functions, gradually improving the proposed algorithms and advancing the optimization functions. As a baseline for developing this thesis, we used the author's previously published algorithms, which are further presented. At this point, we emphasize that such papers were proposed to deal with the PSP. However, one of the goals of this work is to connect and apply them combined with other optimization strategies and analyses. Thus, by an incremental design, we aim to create a search method that can deal with a range of optimization functions, trying to preserve accurate results.

We proposed the development of an adaptive Memetic Algorithm (MA) (MOSCATO, 1989) as a framework-based method with multiple populations, which incorporates concepts of bio-inspired methods and local search (LS) technique (MOSCATO; COTTA, 2019). Following the previously introduced points of interest outlined for this thesis, it aims to efficiently explore the functions' search space, increase the exploitation performance and the diversity maintenance of solutions, and control the metaheuristic parameter values to better deal with the multimodal optimization functions.

MAs simulate the behavior and interactions of individuals (population) based on the "meme" concept of replicating ideas (DAWKINS, 1976). This concept originated from cultural evolution and can be explained as a component of cultural transmission. Complex ideas are divided between agents of a given population who propagate and modify them. Interactions between members of the same community or distinct communities are simulated through global search operations and local refinements that lead to the constant evolution and improvement of the individuals. Still, it is inferred that ideas are the results of search operators, and, as in a cultural environment, good ideas tend to survive. In contrast, bad ones disappear over generations, resulting in a final set of acceptable solutions (NERI; COTTA; MOSCATO, 2012). Due to the enormous complexity addressed by multimodal functions and the PSP, the metaheuristic adopted in this work was motivated from the basic definition of the MAs, which allows great flexibility in the use of global and local optimization heuristics, facilitating the exploration, population diversification, and refinement of the local minima found (MOSCATO; COTTA, 2010).

Besides all concerns regarding metaheuristic design for multimodal functions, our method incorporates the knowledge of 3-D protein structures stored in the PDB to better deal with the PSP. Also, analyzing from a biological perspective, the application of MAs favors the exploration of the protein conformational space through the use of global search strategies. It aims to find distinct structural models while making minor adjustments in the structures found to improve these models correcting eventual conformational errors.

The proposed MA was combined with a modified Artificial Bee Colony (ABC) algorithm (KARABOGA; BASTURK, 2007) used as an exploratory method applied to the MA populations and with the Solis and Wets (SW) algorithm (SOLIS; WETS, 1981) as LS strategy. The ABC is a SI-based metaheuristic that emulates the foraging process of honeybee swarms. It is suitable for multivariate numerical function optimization (AKAY; KARABOGA, 2012). Numerous studies have been published demonstrating its competitiveness with other population-based metaheuristics (KARABOGA; BASTURK, 2007; KARABOGA; BASTURK, 2008; KARABOGA; AKAY, 2009; KARABOGA et al., 2014). Thus, the synergy of concepts between the MA with LS and ABC algorithms seems to be initially reasonable to the problem domain understudy in the sense of search space exploration and refinement.

This study provides an overview of some of the most relevant methods for multimodal optimization in general and also for the PSP problem. The key objective is to develop an adaptive population-based metaheuristic with parameter control strategies, assuring an acceptable exploration, exploitation, and diversity of the model applied to continuous and multimodal benchmark functions and the PSP problem. Hence, the most significant contribution of this work is the design and assessment of an optimization technique, aiming to deal with the points of interest and challenges already mentioned regarding metaheuristics and the PSP.

1.5 Research Objectives

The general objective of this thesis is the study and development of a framework-based metaheuristic for multimodal optimization. The method aims to reach acceptable rates of population diversity and convergence according to the optimization process status, an adequate trade-off between exploration and exploitation through the rugged search spaces, and a suitable parameter setting when applied to distinct benchmarks of continuous functions and the PSP problem.

The goals to be achieved with the development of this work are:

1. To study the most relevant bio-inspired metaheuristics and search techniques in the literature related to the multimodal optimization and its issues, in addition to the state-of-the-art methods for the prediction of 3-D protein structures;
2. To consider the main characteristics of the PSP problem, related to the FM category,

aiming at its modeling as an optimization problem, as well as its constraints and challenges;

3. Based on the survey conducted, to propose a MA for multimodal optimization functions, in order to efficiently deal with these complex landscapes;
4. To implement and validate the proposed approach by tests conducted over well-known benchmark test functions in the continuous optimization domain;
5. To evaluate the obtained results for more general optimization scenarios;
6. Based on the obtained results in the benchmark functions, to adapt the method to deal with the PSP problem and predict approximate solutions;
7. To adapt the metaheuristic to deal with the multimodality and complexities of the PSP and its energy function, aiming to overcome some existing inefficiencies;
8. To evaluate the method performance regarding computational aspects and biological significance of the results;
9. To point out the suitable search strategies regarding each multimodal fitness landscape tested and compare them when analyzing the performance on benchmark and real problems;
10. To validate the proposed algorithm by comparing it with relevant works and state-of-the-art methods in the PSP field (MOULT et al., 2018; ABRIATA et al., 2018).

Finally, it is expected to generate a framework-based MA with different search components and strategies, which can deal with multimodal optimization landscapes, and that intelligently incorporates problem-specific components to predict good-enough native-like protein conformations.

1.6 Thesis Overview

The next chapters of this thesis are organized as follows:

- Chapter 2: The Computational Optimization Background chapter presents a description of computational optimization concepts regarding its definition and according to the scope of this work, which encompasses global and multimodal continuous optimization, optimization strategies, and the main concerns about the search algorithms and the optimization process;
- Chapter 3: The Biological Background chapter provides biological foundation con-

cepts required to understand the protein structure prediction field. The goal of the chapter is to discuss the central notions that comprise the computational prediction of 3-D protein structures, its definition as an optimization problem aiming at the theoretical basis of the PSP problem, as well as the issues related to it;

- Chapter 4: The Related Works chapter presents a review of the most relevant and state-of-the-art metaheuristics used for global and multimodal continuous optimization. The chapter focuses on the literature concerning search strategies to deal with the optimization issues mentioned earlier, such as multimodal search strategies, population diversity and convergence, exploration and exploitation trade-off, and hybrid and parameter control algorithms. The chapter also describes an overview of methods applied to the PSP problem;
- Chapter 5: The Material and Methods chapter presents the algorithms and search strategies used in this work to deal with optimization problems regarding the multimodal continuous domain, as well as the constructive methodology of the proposed methods;
- Chapter 6: The Computational Experiments chapter describes the experiments conducted to analyze the defined optimization case studies and the corresponding algorithms. The chapter also describes the benchmark test functions employed in each scenario of optimization, the algorithms used for comparison, the parameter setting and the metrics applied for evaluation, and the results obtained regarding the performance of the proposed methods facing each one of the case studies;
- Chapter 7: The Conclusions chapter presents the conclusions and final considerations of this work, as well as the future works delineated from the development of this thesis and the obtained results.

2 COMPUTATIONAL OPTIMIZATION BACKGROUND

2.1 Introduction

In this chapter, computational optimization concepts regarding its definition and according to the scope of this work are presented, which encompass global and multi-modal continuous optimization, optimization strategies, and the main concerns about the search algorithms and the optimization process.

An optimization problem is defined as the effort of finding the best possible solution to a specific problem from all feasible candidates (TEZEL; MERT, 2020). Such problems are found in many real-life applications, such as science, engineering, management, and business (SER et al., 2019; HUSSAIN et al., 2019). These problems can be divided into several categories, whether continuous or discrete, constrained or unconstrained, mono or multi-objective, static or dynamic (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). As already introduced, this work is focused on the continuous optimization domain.

2.2 Single Global Continuous Optimization

The global optimization process represents the search for the single best objective variables of a given problem that correspond to either minimum or maximum value of an objective function (TALBI, 2009; KRAMER, 2014). A general global optimization problem can be defined by the tuple (S, f), where S represents the set of feasible solutions (search space), and the relation $f : S \rightarrow \mathbb{R}$ the objective or fitness function to be optimized. The objective function assigns a real value indicating its fitness to every solution $x \in S$ of the search space. The fitness function f allows defining a total order relation between any pair of solutions in the search space. For continuous domain, the search space S is the set $\mathbb{R}$ of continuous values. In most cases, many values have to be optimized at the same time, resulting in an N-dimensional search problem, which defines $S = \mathbb{R}^N$.

Regarding the optimality definition, a global optimal solution $x^* \in \mathbb{R}^N$ has a better fitness $f(x^*)$ than all other solutions of the search space $\mathbb{R}^N$, that is, for a global optimum, it ensures $f(x^*) \leq f(x), \forall x \in \mathbb{R}^N$. Without loss of generality, this definition is made in terms of minimization problems. Maximization problems can be transformed into minimization by inversion of the fitness function $f_{min}(x) = -f_{max}(x)$. Also, a solution x^* with a better fitness $f(x^*) < f(x)$ than the solutions in its neighborhood $x \in \mathbb{R}^N$ with

$\| x - x^* \| < \epsilon$ for an $\epsilon > 0$ is considered a local optimum.

Therefore, the main goal of a global continuous optimization problem is to find a single global optimal solution $x^* \in \mathbb{R}^N$. Many global optima may exist for a given problem, depending on the optimization function. Thus, to get more alternatives, such as in the multimodal optimization (LI et al., 2016), the problem may also be defined as finding all existing global optimal solutions (TALBI, 2009).

2.3 Multimodal Continuous Optimization

It is well-known that numerous problems in many distinct areas present complex objective functions to determine possible solutions (GLIBOVETS; GULAYEVA, 2013). For example, the energy functions used in the prediction process of 3-D protein structures fall into the complex category of multimodal objective functions (HANDL; LOVELL; KNOWLES, 2008; KIM et al., 2009). Multimodal optimization seeks to overcome the complexities imposed by functions multimodality through adaptations in the search algorithms. Its goal is to find all possible optimal or acceptable suboptimal solutions, not just a single solution to the problem (DAS et al., 2011). In literature, the term multimodal optimization also refers to seeking a single global optimum on a multimodal fitness landscape (LI et al., 2016). In this thesis, the optimization goal varies depending on the optimization function and the number of global optima associated with it.

From the previously described definition of a continuous optimization problem, a multimodal continuous optimization represents the relation (S, f), where $S = \mathbb{R}$ is the multimodal search space and $f : S \rightarrow \mathbb{R}$ the multimodal objective function that maps elements of S into a real domain $\mathbb{R}$. Assuming minimization, as $\min f(x), x \in \mathbb{R}^N$, where x is an N-dimensional vector $[x_1, \cdots, x_n]$. In multimodal optimization, a niching method aims to locate all possible $x^* \in \mathbb{R}^N$, not just a single x^*, which holds the minimum possible objective values as $f(x^*) \leq f(x), \forall x \in \mathbb{R}^N$. The fitness values in the neighborhood of an optimal point x^* should be all equal or higher than $f(x^*)$, which minimizes its objective value.

Finding multiple solutions in a single run can improve the metaheuristic performance, given that many points of the search space are optimized at the same time and can be easily changed without affecting the overall performance of the process (GLIBOVETS; GULAYEVA, 2013). Multimodal optimization may increase the probability of a metaheuristic finding global optima since search efforts are not concentrated just in one region of the search space but in different areas. The discovery of many solutions in distinct regions of the search space can also maintain a diverse population, preventing a premature population convergence to local optima (LI et al., 2016; SER et al., 2019).

Single-solution-based algorithms aim to optimize a single solution per run and are commonly used to find only an optimal result of the evaluation function. When algorithms of this type are used in multimodal optimization, they must be applied repeatedly, expecting to find a different solution for each run. In this sense, population-based metaheuristics, such as EAs, have advantages over other more classical search heuristics that are not based on a population of solutions. Ideally, suppose a search algorithm can maintain the diversity of solutions from a satisfactory exploration of the search space at the end of the algorithm's execution. In that case, it is possible to reach multiple good-enough solutions instead of just one (DAS et al., 2011). However, regarding larger search spaces, due to the genetic drift inherent to the population evolution (BELDA et al., 2007), evolutionary metaheuristics also tend to converge to a single global optimum naturally. The genetic drift can lead the entire population to be restricted and spend all the computational effort to optimize only such a point. With this, the preservation of population diversity by the discovery and maintenance of multiple solutions throughout the algorithm's execution configures one of the main challenges in metaheuristics over multimodal optimization.

Solutions' diversity is a critical issue regarding the metaheuristic's ability to explore the search space. In population-based metaheuristics, the population convergence, i.e., the population diversity loss, means that the search process has lost its optimization capacity. As already mentioned, reaching a good balance between preserving the diversity by exploration and refining the local points by exploitation is a common path for population-based algorithms aiming at locating a global optimum (LI et al., 2016).

With this, niching strategies can maintain a more diverse population while locating more than one global optimum. Niching is primarily used to prevent the best population's solution from replacing others with similar objective value but distant objective variables (LI et al., 2016). It is noted that simply preserving a high level of population diversity is not enough for niching since a high population diversity could encompass only random points. This means that a niching algorithm must converge to global optima locally, promoting the formation of distinct subpopulations. So each subpopulation is located around one of the unknown optima (GLIBOVETS; GULAYEVA, 2013).

2.3.1 Niching Strategies

The most common strategies used in multimodal optimization are based on the niching concept (MAHFOUD, 1995; GLIBOVETS; GULAYEVA, 2013), which concerns the attempt to find and preserve multiple solutions around distinct niches or regions of the search space to avoid convergence to a single optimal point. In the case of global optimization with the multimodal objective function, niching may prevent that the algorithm converges prematurely.

Commonly, this niching effect is induced by modifying the selection mechanism of individuals, which considers the fitness function and the distribution of individuals in the solution space (objective variables) or objective space. It is used some distance metric to measure the closeness or differences among individuals in the same or different niches (LI et al., 2016).

Several niching algorithms have been proposed in literature over the years (QU; SUGANTHAN; DAS, 2013), such as Crowding strategy (JONG, 1975; THOMSEN, 2004), Restricted Tournament Selection (RTS) (HARIK, 1995), Fitness Sharing (GOLDBERG; RICHARDSON et al., 1987) and Speciation (LI et al., 2002). Nevertheless, independent of the niching strategy, for multimodal optimization, it is imperative to find global optima by an efficient exploration of the search space and refinement of discovered regions, not to lose them throughout the optimization process. That is the need to distinguish distinct regions of the search space and provide better guidance for the process. A detailed discussion about multimodal algorithms is given in Section 4.6.

2.4 Exploration and Exploitation

Population-based metaheuristic holds a population of individuals and applies specific search equations to find candidates within a feasible search space. It is a consensus among researchers that metaheuristics can reach a better performance when an appropriate balance between exploration and exploitation (Expr/Expt) of solutions is achieved. Also, empirical experiments have shown a strong relationship between the Expr/Expt capacity of a search method and its convergence rate (MORALES-CASTAÑEDA et al., 2020). It means that a good balance can be a factor for the ideal convergence of the metaheuristic. On the other hand, a non-satisfactory ratio can increase the probability of entrapment into local optima (premature convergence) or deteriorate the algorithm's convergence speed. However, according to Morales-Castañeda et al. (MORALES-CASTAÑEDA et al., 2020) and Črepinšek et al. (ČREPINŠEK; LIU; MERNIK, 2013), despite the general agreement on these concepts, there is no clear understanding of what the Expr/Expt trade-off really represents. With this, many measures have been proposed to quantify this balance during the optimization. Most of them are related to the monitoring of the current population diversity.

Thus, the term exploration is related to the solutions' diversification, while exploitation represents the search intensification around promising areas of the search space. Diversification means the ability to visit different regions of the search space, and intensification concerns the method's capacity to reach high-quality solutions within those regions (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). A metaheuristic may address a reasonable balance strategy between these two distinct goals (MORALES-CASTAÑEDA et al., 2020). Search operators and optimization strategies present abilities both of Expr/Expt, and try to reach such a balance. However, it is noted that frequently some of them present a particular specialization in diversification and others toward intensification. Despite this, they tend to get trapped in local optima rather than a global optimum. The main reason is the difficulty to balance such abilities properly by the parameter setting (XU; ZHANG, 2014). An alternative to force the trade-off on metaheuristics could be the design of hybrid algorithms, with specialized search operators in both discovery and refinement solutions (SER et al., 2019).

By characterizing problem difficulties, it was shown that Expr/Expt balance is strictly problem-dependent (XU; ZHANG, 2014). In this sense, considering that metaheuristics can be different in terms of search components and optimization strategies, it is tough to establish an appropriate Expr/Expt ratio that works for all existing methods. It is crucial to understand the mechanisms in these algorithms to devise an efficient search strategy (MORALES-CASTAÑEDA et al., 2020).

For example, in EAs, Expr/Expt is reached by the selection, mutation, and crossover operators (ČREPINŠEK; LIU; MERNIK, 2013). The search components and their parameters are responsible for determining a good Expr/Expt ratio over the search space. Thus, several strategies to deal with the Expr/Expt trade-off in metaheuristics have been proposed. Regarding the search mechanisms, they can encompass adaptive and hybrid approaches. This ratio can be balanced by parameter control in search components for exploration or exploitation and strategies based on population diversity preservation, such as multimodal and niching methods (ČREPINŠEK; LIU; MERNIK, 2013). Besides, an incremental design of metaheuristics may ease clarify the contributions of each component during the search process.

Nonetheless, the parameter control is also problem-dependent, as the optimal parameter set for a particular problem may not be ideal for another (ALETI; MOSER, 2016; PARPINELLI et al., 2019). Still, different parameter values can be the best at different optimization stages, and then the Expr/Expt efforts may also change during the process (ČREPINŠEK; LIU; MERNIK, 2013; LACERDA et al., 2021). The parameter control explicitly controls search components but only implicitly regulates the population diversity, which, in turn, describes Expr/Expt. Therefore, the goal of any search algorithm is to achieve proper diversification to find a good balance between Expr/Expt for different optimization functions. This relationship can be regulated adaptively during the run, addressing parameter control and population diversity issues (ČREPINŠEK; LIU; MERNIK, 2013; XU; ZHANG, 2014). A detailed discussion about parameter control algorithms is given in Section 4.3.

2.5 Population Diversity

As the Expr/Expt ratio, the population diversity is directly related to the convergence rate (SER et al., 2019). At the wrong time, the population diversity loss is considered the primary reason for premature convergence and stagnation (GUPTA; GHAFIR, 2012). Premature convergence means that the metaheuristic's population has reached such a suboptimal state where the search operators can no longer produce new solutions that outperform their current ones in terms of objective values.

In population-based algorithms, the best solutions tend to attract the search process towards them. Consequently, the distance among solutions decreases (diversity loss), whereas the exploitation effort increases, hence the convergence rate. On the other hand, when the distance among solutions increases (diversity gain), the exploration effort prevails (MORALES-CASTAÑEDA et al., 2020). Then, diversity is related to differences among individuals, which can be at the genotype (objective variables) or phenotype (objective values) levels (ČREPINŠEK; LIU; MERNIK, 2013).

Several measures for diversity have been proposed in the literature, aiming to evaluate the differences among solutions of the population (HALIM; ISMAIL; DAS, 2020). However, as diversity measures are problem-dependent, no single measure fits all problems and algorithms. If any measure is used in the optimization process, it is necessary to investigate if positive correlations exist between better performance and the diversity rate (ČREPINŠEK; LIU; MERNIK, 2013). As previously stated, a diverse population is mandatory for exploration to avoid premature convergence. For example, a diverse population can deal with multimodal functions, simultaneously exploring several peaks in the fitness landscape. On the contrary, high diversity at all phases of an optimization process may not be ideal since there exist stages where exploitative behavior is needed (ČREPINŠEK; LIU; MERNIK, 2013).

Regarding the continuous optimization domain, diversity measures can be classified as distance-based, entropy-based, and ancestry-based distances (ČREPINŠEK; LIU; MERNIK, 2013). The distance-based measure is the most widely used type of diversity measure. Several distances are considered, including Hamming, Euclidean, cosine distance of similarity, edit distance, and distance to the average point. Entropy-based is a promising measure representing the population disorder, where higher entropy means higher diversity. Moreover, the ancestry-based measure is calculated by comparing the current population with those populations of previous generations, considering ancestries or history. Among these types, some of them are described below (ČREPINŠEK; LIU; MERNIK, 2013; HALIM; ISMAIL; DAS, 2020).

The dimension-wise diversity measure can be considered to calculate the distance variation between solutions (CHENG et al., 2014; MORALES-CASTAÑEDA et al., 2020). Such measure is distance-based and can be defined according to the Equation 2.1.

$$Div_{dimension}(P) = \frac{1}{D}\sum_{j=1}^{D}\frac{1}{N}\sum_{i=1}^{N} \mid x_{ij} - median(x_j) \mid \tag{2.1}$$

Where $median(x_j)$ represents the median of the j-th dimension in the whole population P. x_{ij} is the j-th dimension of solution i. N is the population size and D is the problem dimension. The diversity of each dimension is calculated as the distance between the j-th dimension of every solution and the median of that dimension, averaged. The use of the median value avoids inconsistencies by using a reference element. However, one possible shortcoming of this measure is the smoothing effect produced by the average combination of all dimensions, where small changes in diversity are partially smoothed (MORALES-CASTAÑEDA et al., 2020). The diversity of the whole population $Div_{dimension}$ is the average of every diversity in each dimension. Regarding the equation result $Div_{dimension}$, higher $Div_{dimension}$ values means higher population diversity rate.

Another similar measure to the above aims to measure the diversification of solutions around the population center (KRINK; VESTERSTROM; RIGET, 2002; HALIM; ISMAIL; DAS, 2020). The measure is based on the average distance of solutions over the center of population P, calculated as the average of each dimension variables, as follows:

$$Div_{center}(P) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_{j=1}^{D} \mid x_{ij} - \bar{x}_j \mid} \tag{2.2}$$

Where N is the population size, D is the problem dimension, x_{ij} is the j-th variable value of the i-th solution, and $\bar{x}_j$ is the j-th value of the average point of solutions. A lower Div_{center} value indicates lower diversity. Hence convergence around the population center, while higher values indicate a higher dispersion of solutions away from the population center.

A similar diversity measure can be expressed by Equation 2.3. This measure tends to amplify greater distances between pairs of solutions. However, a potential drawback is that the squaring will also amplify the distances produced by outliers.

$$Div^2_{center}(P) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{D}\sum_{j=1}^{D}(x_{ij} - \bar{x_j})^2} \tag{2.3}$$

Another possible measure is an entropy-based measure (TANG et al., 2015; HALIM; ISMAIL; DAS, 2020) that divides the phenotype search space into Q regions of equal size with Z_i solutions in each region, considering a population P of size N. The probability of solutions situated in the i-th region is determined by Z_i/N. The population-entropy diversity measure describes where solutions are distributed among the various domains in search space considering their fitness values. Such diversity distribution can be achieved by the identification of evolutionary states and is given by the following steps (TANG et al., 2015):

1. Assuming minimization of objective function $f(x)$, and that Fp is an array with size N, where each Fp_i represents a fitness value corresponding to the i-th solution of the population, let $f_{max} = \max(Fp)$ and $f_{min} = (Fp_{best})$;
2. Consider an interval $[f_{min}, f_{max}]$;
3. Divide $[f_{min}, f_{max}]$ into N intervals of equal length.
4. Assign each solution considering its fitness to the correspondent interval and find the specific number n_i for each of those intervals;

5. Diversity of the population distribution is then given by Equation 2.4, where $Q = N$.

$$Div_{entropy}(P) = -\sum_{j=1}^{Q} \frac{Z_i}{N} log_e \left(\frac{Z_i}{N} \right) \tag{2.4}$$

Regarding the equation result $Div_{entropy}$, high value of $Div_{entropy}$ means high population diversity rate.

The population diversity can also be measured in terms of the differences among the fitness of the population's solutions (SABAR; ALETI, 2017). The degree of population diversity is directly estimated using the objective values of solutions regarding the average, best and worst fitness values of the current population, according to Equation 2.5.

$$Div_{fitness}(P) = \left| \frac{f_{avg} - f_{best}}{f_{worst} - f_{best}} \right| \tag{2.5}$$

Where f_{avg}, f_{best} and f_{worst} represent the average, best and worst fitness values of the population, respectively. Assuming minimization problem, higher values of $Div_{fitness}$ means higher population diversity rates.

Also, the coefficient of variation (CV) of a population can be used as a diversity measure (CORRÊA; DORN, 2020), which calculates the relative variability to estimate the sample diversity degree (BEDEIAN; MOSSHOLDER, 2000). The CV represents the ratio of the standard deviation $\sigma(P)$ of a given population P to the population mean $\bar{P}$, according to the Equation 2.6. Lower CV values indicate that the population tends to be similar.

$$CV(P) = \frac{\sigma(P)}{\bar{P}} \tag{2.6}$$

Finally, the most used strategy for reaching a satisfactory Expr/Expt balance is maintaining a diverse population. Although a diverse population is mandatory for a good Expr/Expt trade-off, it is not guaranteed. A successful optimization process also depends on the Expr/Expt abilities of the metaheuristic's components and parameterization.

2.6 Exploration and Exploitation Trade-off Definition

Investigations around the efficiency of metaheuristics have attracted attention, notably concerning the potential of a good balance between Expr/Expt. Many strategies have been proposed to quantify this trade-off numerically, such as measuring different diversity metrics over a population of solutions, which might indicate the level of diversification of the metaheuristic along with generations. Moreover, diversity has been identified as a guidance for the metaheuristic's success. However, it is also known that some loss of diversity is required for some search algorithms to converge properly (SER et al., 2019). According to Črepinšek et al. (ČREPINŠEK; LIU; MERNIK, 2013), one can define the Expr/Expt trade-off as follows.

Adopting any diversity measure, let $d(\cdot,\cdot)$ denotes the diversity between two solutions of a population P, where d is a diversity function $d : PxP \rightarrow \mathbb{R}$. Then, the similarity to the closest neighbor (SCN) can be used to distinguish exploration from exploitation. When a new solution sol_{new} is created, a similarity measure to the closest neighbor (SCN) can be defined, but not strictly, regarding: (*i*) the similarity to its parents; (*ii*) the similarity to the most similar solution within the entire population P; and (*iii*) the similarity to the most similar solution in the subpopulation or niche $P' \wedge (P' \subset P)$. Following the multimodal principles and considering the third point mentioned above, SCN is given by Equation 2.7.

$$SCN(sol_{new}, P') = \min_{\substack{sol \in P' \wedge (P' \subset P) \\ sol_{new} \neq sol}} d(sol_{new}, sol) \tag{2.7}$$

Exploration occurs when $SCN(sol_{new}, P') > X$, where X denotes a threshold value that defines the neighborhood's limit of the closest neighbor and is problem-dependent. Then, the exploration process explores points that are outside of the current neighborhood of the closest neighbor. Contrarily, exploitation occurs when $SCN(sol_{new}, P') \leq X$.

Another definition of Expr/Expt trade-off is presented in Morales-Castañeda et al. (MORALES-CASTAÑEDA et al., 2020), and represent it as the percentage of Expr/Expt invested by a given metaheuristic. Given any diversity measure, the Expr/Expt ratio is defined as follows.

$$Expr\% = \left(\frac{Div}{Div_{max}}\right) \times 100 \tag{2.8}$$

$$Expt\% = \left(\frac{\mid Div - Div_{max} \mid}{Div_{max}} \right) \times 100 \tag{2.9}$$

Where Div is the diversity rate of the current iteration and Div_{max} is the maximum diversity value found in the entire optimization process. The percentage of exploration $Expr\%$ describes the exploration effort as the relationship between the diversity in each iteration and the maximum reached diversity. The percentage of exploitation $Expt\%$ is calculated as the complementary percentage to $Expr\%$ and describes the exploitation effort. It is noted that both $Expr\%$ and $Expt\%$ are mutually conflicting and complementary.

Therefore, such definitions of Expr/Expt can be used to evaluate the relationship between the population diversity, the Expr/Expt trade-off, and the metaheuristic's performance. Thus, if positive correlations exist between better performance and the diversity measure, it is also possible to relate such correlation to the current Expr/Expt balance.

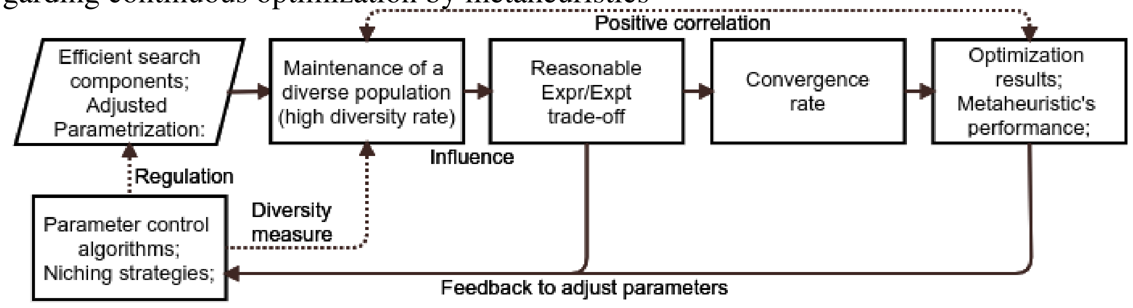
2.7 Connecting the Points of Interest

In essence, since a proper Expr/Expt trade-off can improve the metaheuristic's performance, the population diversity is strictly related to this issue. Assuming that the trade-off is a concept used to understand better the behavior of metaheuristic search mechanisms and their efforts to explore and refine the solutions. Then, the metaheuristic search components are the real responsible for the success of the algorithm. Their operation and parameter setting define how well the algorithm explores and refines the search space, establishing the balance between them based on its configuration. The search components and their parameter set define the population diversity rate, which, in turn, describes the Expr/Expt balance and defines the convergence rate. Finally, the convergence rate, and if the method has converged ideally to global optima, determine the success of the optimization process.

Therefore, based on this cause and effect relationship, there are signs that the metaheuristic components and parameterization are the most important optimization aspects. They can regulate all the rest of the established factors, which encompass the diversity/convergence rates, Expr/Expt trade-off, and the final optimization results. Thus, if we adopt improper search algorithms and parameter values, then the diversification of solutions may be impacted negatively. It can cause premature or non-convergence and, consequently, adverse outcomes.

To tackle an optimization problem, it is necessary to choose the metaheuristic appropriately, incorporate the strategies suitable for the problem under study, and define an acceptable parameter setting. By the monitoring of population diversification, it is possible to regulate the Expr/Expt efforts. An acceptable diversity rate has the potential to guide the algorithm's convergence to success as long as the search mechanisms and parameter settings are equally efficient, e.g., based on population diversity preservation. If the diversity is not as good as expected, the components and the parameterization can be changed throughout the execution. In this sense, it is necessary to address the parameter control problem, which was already introduced as one of the points of interest of this work and will be further examined (Section 4.3). Adapting the parameterization during the optimization process based on the metaheuristic's feedback regarding the current level of population diversification can adjust parameter setting errors and reduce search bias, leading to an acceptable trade-off between Expr/Expt. This may lead to a satisfactory convergence and better optimization results. We emphasize that the relationship between components/parameter set with population diversification can be seen as a "two-way channel" since the metaheuristic's configuration defines the diversity level. The diversity can change the configuration via parameter control during the process. It is a complex cause and effect relationship where those involved can be the cause and also the effect. Figure 2.1 illustrates the relationship between the points of interest discussed in this chapter and outlined for this thesis concerning continuous optimization aspects.

Figure 2.1: Relationship between the points of interest outlined for this thesis work regarding continuous optimization by metaheuristics


Positive correlation
Efficient search components; Adjusted Parametrization;
Maintenance of a diverse population (high diversity rate)
Reasonable Expr/Expt trade-off
Convergence rate
Optimization results; Metaheuristic's performance;
Influence
Regulation
Diversity measure
Parameter control algorithms; Niching strategies;
Feedback to adjust parameters

Source: From the author (2022).

Finally, this section was written based on the previously explored concepts about the efficient exploration of the search space, Expr/Expt trade-off, population diversity, and convergence factors. It is an attempt to establish a connection among them and guide the design of our metaheuristics.

2.8 Strategies for Achieving Diversity, Exploration and Exploitation

From the Expr/Expt perspective, a higher diversity indicates that the search algorithm is in the exploration stage, while a decrease in diversity indicates an exploitation search. Such relationship between exploration and diversity is defined by the search components and strategies which preserve the population diversity. According to Črepinšek et al. (ČREPINŠEK; LIU; MERNIK, 2013), search methods can address the Expr/Expt balance, but are not limited to, through: (*i*) diversity preservation; (*ii*) diversity control; and (*iii*) other more direct strategies. The first two groups are primarily focused on population diversity and only implicitly address Expr/Expt issue. The latter deals with the trade-off more directly.

The first group aims to improve the Expr/Expt ratio assuming that the search techniques can preserve diversity per se and reach a good Expr/Expt balance. Several methods have been proposed for diversity preservation. They can be categorized as niching (LI et al., 2016) and non-niching strategies (TOFFOLO; BENINI, 2003). Both of them can maintain a diverse population, but niching methods are also able to locate multiple optimal solutions. It is noted that niching methods can be considered more robust diversification methods. Therefore, besides the niching methods already mentioned in Section 2.3.1, some popular non-niching strategies used to maintain the population diversity in evolutionary metaheuristics comprise: (*i*) population-based, where diversity is achieved by varying the population size, duplicate elimination of similar solutions, infusion or reinitialization techniques, external archives, or migration between subpopulations; (*ii*) selection-based, where diversity is achieved by changing selection pressure or replacement restrictions between new solutions and the current population; (*iii*) crossover or mutation-based, where mating restrictions or disruptive operators achieve diversity; and (*iv*) hybrid approaches, where ensembles of these strategies achieve diversity.

The second group deals with the Expr/Expt efforts by the diversity control. It considers population diversity, objectives values of population, or fitness improvements as feedback to guide an optimization process towards exploration or exploitation. In this category, methods can be classified according to the strategy or operator responsible for diversification. Some of them include: (*i*) diversity is controlled by selection mechanism, where survival probability can be considered in terms of population diversity; (*ii*) diversity is regulated by crossover and mutation operators, which is the most adopted category and intends to increase or decrease the probability of operators after the computation of population diversity, objective values, or fitness improvements. In this group, methods differ from each other mainly on how diversity is calculated, i.e., explicitly by diversity measures or implicitly by fitness or improvements; and (*iii*) diversity is regulated by population changes, where the population size or the population itself can be changed, e.g., in a reinitialization process.

The third group comprises strategies that try to control the Expr/Expt ratio more directly. Such methods suggest direct control between Expr/Expt, and these two stages are identified and interleaved. The strategies can be divided into three groups. The first group uses distinct subpopulations for a specific phase of exploration or exploitation. Subpopulations are used to delimit exploration from exploitation, and some of them are exclusively used for a particular search stage. The second group uses the same population of solutions, but different triggers cause alternation between Expr/Expt. The last one considers an ancestry tree-based approach for explicitly measuring Expr/Expt. The evolution of a population over the optimization process is recorded on an ancestry tree, where diversity measures can be applied to find similarities between solutions on the ancestry tree and the current ones.

Finally, we emphasize that the diversity and the Expr/Expt balance can be addressed by an ensemble of optimization strategies. The concept of ensemble or hybridization in optimization encompasses the combination of multiple search strategies, subpopulations, algorithms, rules for solution selection, operators, and parameterization to deal with an optimization problem. This concept is based on the principle that a hybrid method may reach better results than a single strategy to a given problem (SER et al., 2019).

2.9 Final Remarks

This chapter described computational optimization concepts regarding its theoretical definition as global and multimodal continuous optimization. Also, it provided a characterization of the main issues about metaheuristics and optimization problems, which included: (*i*) a description and formalization of the Expr/Expt trade-off; (*ii*) concerns about population diversity and convergence rate, as well as diversity measures; (*iii*) an attempt to establish a connection among these issues to guide this thesis; and (*iv*) search strategies and methods for achieving population diversity and Expr/Expt of the search space.

The goal of this thesis is the development of a population-based metaheuristic applied to global and multimodal continuous benchmark functions and the PSP problem. In this sense, the next chapter presents the biological background necessary to understand the protein structure prediction problem and how it can be formalized as a computational optimization problem.

3 BIOLOGICAL BACKGROUND

3.1 Introduction

The goal of the chapter is to discuss the central notions that comprise the computational prediction of 3-D protein structures, its definition as an optimization problem aiming at the theoretical basis of the PSP problem, and the issues related to it.

3.2 Physicochemical Protein Composition

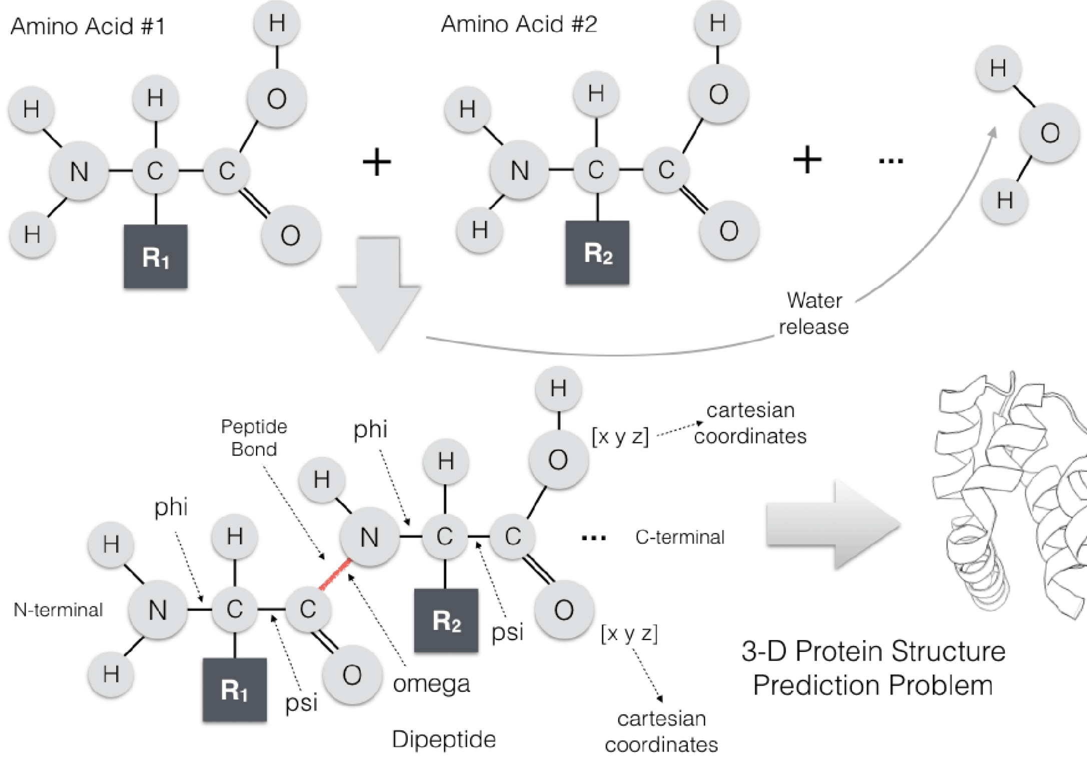
From a structural perspective, proteins or polypeptides are folded chains of amino acids (aa). An aa is a small molecule consisting of an amino group ($\mathtt{NH_2}$), a carboxyl group ($\mathtt{COOH}$), and a hydrogen atom ($\mathtt{H}$) attached to the central alpha carbon ($\mathtt{C}\alpha$), as illustrated in Figure 3.1. In addition, each aa has also an organic group $\mathtt{R}$ (side chain) attached to the backbone $\mathtt{C}_\alpha$. The $\mathtt{R}$ group is responsible for differentiating one aa from another and gives the specific physicochemical properties of each aa (LODISH et al., 2007). There are 20 types of common amino acids, whereas each one has its characteristics. The aa side chains may differ in size, electric charge, and polarity.

Peptides are molecules composed of two or more amino acids linked through chemical chains, known as peptide bonds (Figure 3.1). The peptide bond ($\mathtt{C-N}$) is formed when the carboxyl group of an aa reacts to the amino group of another aa, and thus releases a water molecule ($\mathtt{H_2O}$). Two or more chained amino acids are referred to as peptides, and larger peptides are polypeptides or proteins (CREIGHTON, 1990; BRADLEY; MISURA; BAKER, 2005). Each protein is defined by a unique linear sequence of amino acids responsible for determining its conformation. This conformation or folding provides the protein-specific biochemical properties, which determine its role in the organism (LESK, 2010).

3.3 Levels of Structural Protein Abstraction

Proteins can be divided into four levels of structural abstraction to ease the structures' understanding (BRANDEN; TOOZE, 1999; LODISH et al., 2007): (*i*) primary structure (PS); (*ii*) secondary structure (SS); (*iii*) tertiary structure (TS); and (*iv*) quater-

Figure 3.1: Chemical representation of two amino acids and a schematic model of the peptide formation. The carboxyl group (COOH) of one aa (1) reacts with the amino group (NH_2) of another aa (2), thus releasing a water molecule (H_2O) and creating a peptide bond (C−N). N represents nitrogen atoms; C denotes carbon atoms; O represents oxygen particles, and H are hydrogen atoms



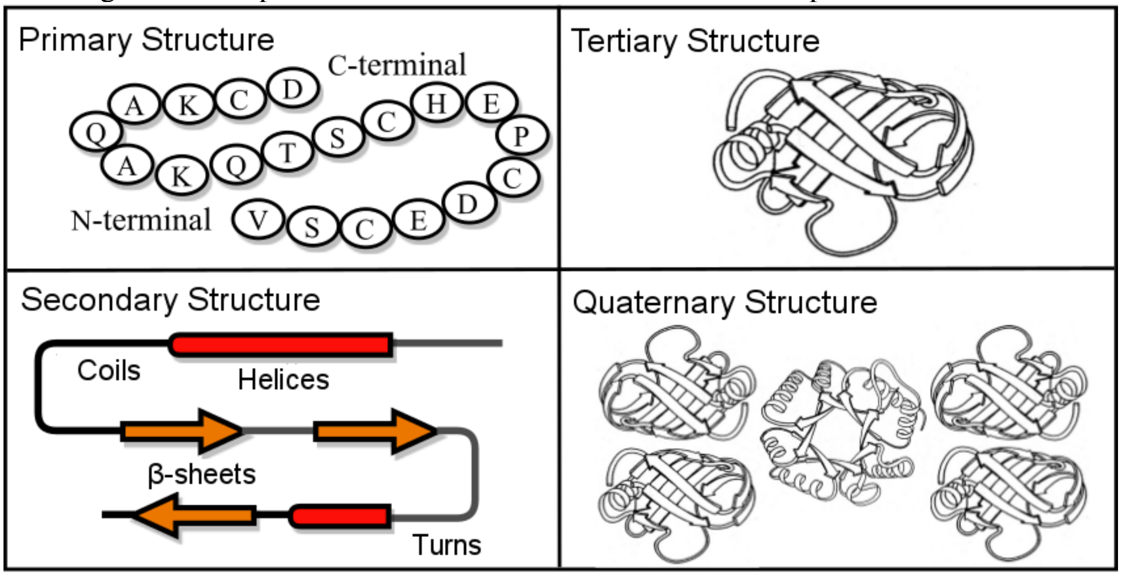
Source: From Corrêa and Dorn (CORRÊA; DORN, 2020).

nary structure (QS). Figure 3.2 illustrates the four levels of structural protein abstraction.

The PS describes only the aa sequence in linear order (BRANDEN; TOOZE, 1999). The SS is defined by the stable aa arrays primarily determined by the presence of hydrogen bond patterns formed through interactions between H and O or N atoms. Due to the high force intensity in molecular interactions of this nature, these structures are responsible for ensuring the protein structure conformational stability in the 3-D space.

The SS can be classified into helices (PAULING; COREY; BRANSON, 1951), β-sheets (PAULING; COREY, 1951), and loops (turns and coils). The helices and sheets structures are more stable conformations and therefore are called regular structures. Helices are stabilized by hydrogen bonds between the N atom of a peptide bond or the O atom of the carboxyl group of the third (3^{10}-helix), fourth (α-helix) or fifth (π-helix) aa of the N-terminal region. β-sheets are chains of extended amino acids combined with other neighboring chains in parallel or antiparallel. The amino and carboxyl groups of

Figure 3.2: Representation of the four levels of structural protein abstraction



Source: From Corrêa and Dorn (CORRÊA; DORN, 2020).

the peptide bonds come closer in the same plane to allow hydrogen bonds between adjacent polypeptide chains. Turns and coils represent highly flexible structures. They are originated in regions where the protein changes its conformation, which means that they occur between regular SSs. These are known as disordered regions, as they present a high flexibility degree due to the greater exposure to the solvent. For this reason, they are more difficult prediction conformations (SCHEEF; FINK, 2009).

The protein TS is related to its 3-D topology, defined by the SS arrangements and connections, besides their positioning in the 3-D space. It is also called the native or functional protein structure. The conformation is constituted and stabilized by several thermodynamic factors, such as hydrogen bonds, hydrophobic and electrostatic interactions, attraction and repulsion *van der* Waals forces, among others (ANFINSEN, 1973).

The QS represents the combination and the 3-D arrangement of two or more polypeptide chains (TS) of the protein. This structure is influenced by the same forces that determine the SSs and TSs (LODISH et al., 2007).

The comprehension of protein folding allows more direct investigations about the biological processes, with greater resolutions and details, despite the complexity involved in those processes. According to the "sequence-to-structure-to-function" paradigm, specific proteins only assume their biological functions when folded in a single and stable conformation (ANFINSEN, 1973). However, it is known that not all functions performed by proteins are directly associated with a native and stable state (TOMPA, 2002;

DUNKER et al., 2008b). In some cases, proteins must remain disordered to perform their functions correctly (GUNASEKARAN et al., 2003). These proteins are called intrinsically disordered proteins (IDP) (DUNKER et al., 2001) and comprise about 30% of known protein sequences. Despite the IDPs, a critical aspect to explain the function of a given protein involves the analysis of complex molecular interactions present in its conformation. These interactions can be intramolecular (ionic, covalent, metallic bonds) or intermolecular (hydrogen bonds and other non-covalent bonds, such as electrostatic and *van der* Waals) forces. Thus, the knowledge concerning the 3-D structure of polypeptides provides researchers with valuable information about the protein's role in the organism (BRANDEN; TOOZE, 1999; LASKOWSKI; WATSON; THORNTON, 2005).

3.4 Structural Protein Classes

It is well-known that most proteins have structural similarities, and in many cases, they also share the same evolutionary origins (FOX; BRENNER; CHANDONIA, 2015). The protein classification into different structural classes, considering the SS arrangements and the conformations adopted, can provide detailed descriptions of the relationships among proteins with known 3-D structures. These classes represent several intermolecular interactions, which originate from different structural SS arrangements and 3-D topologies.

The Structural Classification of Protein database[1] (SCOP) (CONTE et al., 2000) aims to classify structural protein conformations, considering the similarities of aa sequences and structures. SCOP was developed to provide detailed categorizations regarding protein structures of the PDB. These resources provide extensive reviews about different protein conformations, besides information about proteins structurally similar to any other protein already classified (FOX; BRENNER; CHANDONIA, 2015). According to the SCOP, proteins can be classified into different hierarchical levels. The highest level composes the division by classes, aiming to categorize them regarding SS types and arranged arrays. The other hierarchy levels represent more specific categories, such as folding patterns, superfamilies, and families.

In this work, we have defined structural protein classes similar to those presented in the SCOP project. It has aimed to find more specific information about the target

[1]<scop.mrc-lmb.cam.ac.uk/scop/>

proteins under study. Proteins were classified according to their SS composition and arrangements (CHOU; ZHANG, 1995). We note that a more simplified categorization was adopted, which tends to be more general but easier to obtain. The predominance values of SS follow the guidelines delineated by the SCOP and also in the work (CHOU; ZHANG, 1995). In this way, proteins were classified into four distinct classes:

- Class of irregular regions: comprises structures that have more than 80% of turns or coils in their SS composition;
- Class of β-sheets: encompasses proteins that have the predominance of more than 60% of β-sheets in their SS;
- Class of helices: covers structures with more than 60% of helices in their SS; and
- Hybrid class: comprises arrangements that do not fit into any previous classes, i.e., they present a combination of the three SS types.

3.5 Computational Representation of Protein Structures

Proteins can assume diverse conformations in a 3-D space, and the structure of a given protein, defined by the aa sequence, spontaneously folds in nature during or after biosynthesis. The relationship between the amino acid sequence of a protein and its 3-D structure was demonstrated, for the first time, by experiments carried out in the works of Anfinsen et al. (ANFINSEN et al., 1961; ANFINSEN, 1973). It is characterized as a complex process dependent on many factors, such as solvation elements, salt concentration, and temperature. Therefore, the computational representation of 3-D protein structures is a challenging task due to the difficulty in representing the atomic structure and molecular interactions, as well as simulating all the factors responsible for the protein stabilization, as it occurs in nature or even in structural determination performed by experimental methods (ANFINSEN, 1973).

The computational configuration of a protein model is associated with the detail level used to describe its 3-D structure, since the higher the detail level used to describe the model, the higher is the ability to express the protein as in nature, and consequently, the greater the computational complexity involved (MIRNY; SHAKHNOVICH, 2001; CORRÊA; DORN, 2017). More detailed representations encompass all the protein atoms and describe the solvent molecules necessary to the folding process, while others, more simplified ones, abstract some concepts to reduce the computational complexity. However, the simpler the computational representation used, the more significant the information loss and representativity compared to real proteins (MIRNY; SHAKHNOVICH, 2001).

The geometric representation of structures is one of the essential components in computational methods to predict the 3-D protein structures. It is also responsible for reducing or increasing the search space complexity. Thus, overly detailed models, which seek to represent most of the protein atoms and other molecules involved in the folding process, end up rendering the representation computationally expensive. Thus, more simplified representations are often preferable when applied to the PSP (CHIVIAN et al., 2003; CORRÊA; DORN, 2017).

Commonly, two computational representations appear in the literature. The first model represents the 3-D protein structure through the Cartesian position (x, y, z) of the aa atoms. In this case, a polypeptide chain can be described as a set of atoms at arranged in the 3-D space, where $\{at \mid at \in \mathbb{R}^3\}$. The second model represents the protein structure by its dihedral angles, compounded by the chain of amino acids and peptide bonds. This representation is based on the fact that the bond lengths are almost always constant in a polypeptide chain, which enables the reconstruction of the model to represent all the protein atoms (NEUMAIER, 1997; MIRNY; SHAKHNOVICH, 2001). So the dihedral angles serve as the basis for the atoms' positioning and the model reconstruction. A frequently adopted variation of the second model is the centroid-based protein representation (ROHL et al., 2004). In such representation, the main chain remains fully atomic. However, the representation of each aa side chain is simplified to a single pseudo-atom arranged in the side chain center of mass. The centroid-based representation simplifies the side chain complexity whereas keeping the overall protein folding by preserving the backbone integrity (LEAVER-FAY et al., 2011).

The main advantage of using torsion angles, compared to the Cartesian model, is the ability to reduce the degree of freedom of the structural model, reducing conformations and complexity. However, the main disadvantage of such representation is that a slight change in a dihedral angle can cause drastic changes in the rest of the 3-D structure. In Cartesian representations, point changes in atoms have little effect on the overall model structure. There are also other simplified representations, such as the lattice models (KOLINSKI; SKOLNICK, 2004) and the off-lattice AB toy model (STILLINGER; HEAD-GORDON; HIRSHFELD, 1993).

In this thesis, we adopted the computational structural representation based on the torsion angles of the polypeptide chains to reduce the complexity of the all-atom representation while keeping a certain degree of accuracy compared to real protein structures.

3.6 PSP Problem Definition

This section presents the stereochemistry of protein amino acid bonds and the set of dihedral angles formed from them. From this set of dihedral angles, we intend to model the PSP problem computationally.

3.6.1 Stereochemistry

The specific characteristics of peptide bonds between amino acids have significant implications in protein conformations. The peptide bond ($\texttt{C}-\texttt{N}$), responsible for the dihedral angle Omega (ω) formation, has a double partial bond and tends to be planar with two allowed states: *trans*, $\omega = 180°$ (usually) and *cis*, $\omega = 0°$ (rarely), with little or no rotation around its axis. Free rotations are allowed around the $\texttt{N}-\texttt{C}_\alpha$ and $\texttt{C}_\alpha-\texttt{C}$ bonds (BRADLEY; MISURA; BAKER, 2005). These bonds represent the dihedral angles Phi (ϕ) and Psi (ψ), respectively. The angles can be freely rotated in the continuous range $[-180°, 180°]$ and are considered the main responsible for the conformation adopted by the protein backbone. However, the free rotation around ϕ and ψ is limited by stereochemical constraints among the protein main and side chains (SCHEEF; FINK, 2009). As a consequence, the conformation of a specific protein becomes dependent on the aa chemical properties.

Similar to the main chain, the side chain also has dihedral angles, called angles Chi (χ). The aa side chain conformation contributes to the protein stabilization and packing (LEVITT et al., 1997). The number of side chain angles χ depends on the aa type, ranging from 0 to 4 angles with rotational freedom varying from -180° to +180°.

Therefore, the set of angles ϕ, ψ and ω of all protein's amino acids defines the backbone conformation, as the combination of angles χ of each aa configures the protein side chain (HOVMÖLLER; ZHOU; OHLSON, 2002; LIGABUE-BRAUN et al., 2018).

3.6.2 Computational Modeling of the PSP Problem

According to the stereochemistry of amino acids, it is known that from the complete set of all dihedral angles that define the protein conformation, it is possible to reconstruct the computational model to represent all the atoms since the bond lengths are almost always constant in a polypeptide chain. Thus, the computational representation by torsion angles becomes feasible.

Thereby, the 3-D structure of a protein P with n amino acids can be computationally defined only by assigning the main and side chain dihedral angles to the amino acids that encompass the protein (Equation 3.1) since the bond lengths between the atoms are little variable.

$$P = [aa_1, aa_2, \cdots, aa_{n-1}, aa_n] \tag{3.1}$$

$$aa_i = [\phi_i, \psi_i, \omega_i, \chi_{i_{(0 \cdots 3)}}] \tag{3.2}$$

For the main chain representation of the protein P, the model with n amino acids has $3n$ freedom degrees or objective variables to be optimized, but noting that ω angles have little or no variation: trans, $\omega \approx 180^{\circ}$ (usually) and cis, $\omega \approx 0^{\circ}$ (rarely) (BRANDEN; TOOZE, 1999). Then, considering the side chain dihedral angles, the cardinality of the angle set of P (dimensionality of variables) is determined by the Equation 3.3. We note that the ϕ angle of the `N`-terminal region of the main chain and the ψ angle of the `C`-terminal region of the polypeptide are disregarded because they are nonexistent.

$$|P| = 3n(\sum_{1}^{n} |\chi_i|) - 2 \tag{3.3}$$

Nonetheless, in this thesis, the computational representation adopted in the optimization processes is based only on the backbone dihedral angles, disregarding the side chain angles. As mentioned earlier, such representation is known as centroid-based and focuses on the target protein's overall folding. This representation maintains the main chain fully atomic. However, the representation of each aa side chain is simplified to a single pseudo-atom arranged in the side chain center of mass. The centroid-based representation simplifies the side chain complexity whereas keeping the overall protein folding by preserving the backbone integrity. Rewriting the Equation 3.1, in centroid-based models, the 3-D structure of a protein P with n amino acids can be defined only by assigning the main dihedral angles to the amino acids that encompass the protein, according to the

Equation 3.4.

$$P = [aa_1, aa_2, \cdots, aa_{n-1}, aa_n] \tag{3.4}$$

$$aa_i = [\phi_i, \psi_i, \omega_i] \tag{3.5}$$

With this, the PSP problem can be mathematically described as an optimization problem (LEUNG; WANG, 2001). Considering $f(x)$ as the objective function used in the solution evaluation, such that $f(x)$ must be minimized in relation to the range of real numbers defined by $l \leq x \leq u$, where $x = P = [aa_1, aa_2, \cdots, aa_{n-1}, aa_n]$ (Equation 3.1) is a vector of objective variables in space $\mathbb{R}^{|P|}$, as each x is also a vector of objective variables that encompasses the angular values of a given aa (Equation 3.2). l and u define the solution space allowed to each variable, where $l = -180$º and $u = 180$º. The interval $[l, u]$ is common to all variables (dihedral angles) of x.

So we note that in this work, the computational representation used in the optimization processes is based on the dihedral angles of the protein's main chain. However, the model evaluations are performed using the Cartesian representation. We adopted the centroid-based objective function of Rosetta[2], and the model conversion between the dihedral angle representation to the atomic coordinate is done by the own Rosetta's energy function implementation (ROHL et al., 2004; CHAUDHURY; LYSKOV; GRAY, 2010; KAUFMANN et al., 2010).

3.7 Objective Function for the PSP Problem

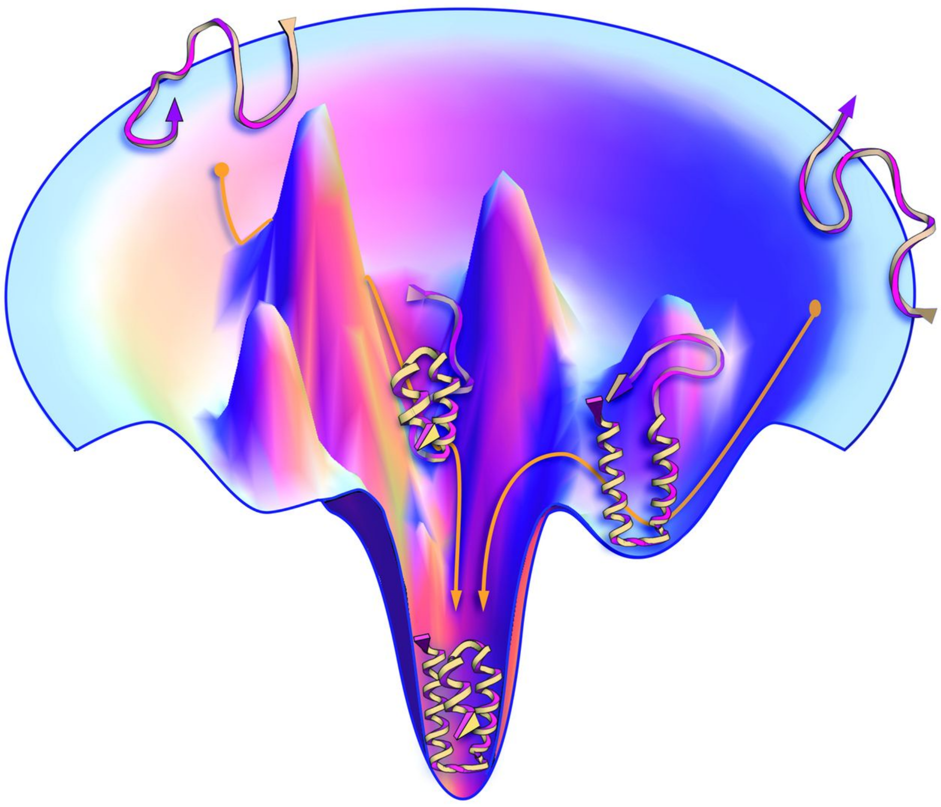
Predicting protein structures involves the generation of several structural solutions to find the closest model to the native protein structure, which represents the one with the lowest potential energy (DILL; MACCALLUM, 2012; FARAGGI; KLOCZKOWSKI, 2014). Energy functions are used in prediction processes to estimate the folding condition of a given model, considering the distribution of its energy value on the energy landscape. They are used as minimization functions and should have the ability to differentiate between more or less stable protein configurations since, theoretically, conformations around a native state must reflect global minimum regions of their free energy (ANFINSEN, 1973). It is said "theoretically" as energy functions used in molecular modeling processes have great difficulties representing real protein structures in nature due to the

[2]<www.rosettacommons.org>

enormous complexity of such systems (KIM et al., 2009).

Energy functions used in the PSP problem comprise the category of multimodal objective functions (Section 2.3), characterized by the highly rough landscape, which gives rise to several valleys seen as local and global optima of the function (HANDL; LOVELL; KNOWLES, 2008), as shown in Figure 3.3. For the same input data, the same multimodal function can present two or more distinct solutions located in different regions of the search space but with similar fitness value (GLIBOVETS; GULAYEVA, 2013).

Figure 3.3: Multimodal landscape with several valleys seen as local and global optima of the energy function



Source: From Dill and MacCallum (DILL; MACCALLUM, 2012).

Concerning the problem, the same energy value may represent distinct conformations for the same target protein, making it difficult to differentiate between models from two different local minima. Also, optimal points tend not to reflect conformations around the polypeptide native state (KIM et al., 2009). Also, it is emphasized the existence of IDPs, which are protein structures that do not count with stable states or are disordered under certain physiological conditions (DUNKER et al., 2008a), where a single global minimum can not represent the native and functional state.

Generally, energy functions designed to the PSP represent potential functions empirically derived from experimentally determined structures from the PDB (HAO; SCHERAGAT, 1999; LAZARIDIS; KARPLUS, 2000). Some of them also incorporate terms from molecular mechanics, which model the forces responsible for determining the protein conformations through physically parameterized functional formats, considering data from small molecules or based on quantum mechanics calculations (JORGENSEN; TIRADO-RIVES, 2005). More generic prototypes of potential energy functions, given by Equation 3.6, can be expressed by linear summation, weighted or not, of some energy terms representing the forces that determine macromolecular conformations, such as bond angles and bond lengths, dihedral angle values, *van der* Waals forces, electrostatic interactions, hydrogen bonds, implicit solvation components, etc. (HANDL; LOVELL; KNOWLES, 2008; MACKERREL, 2010).

$$E_f = E_b + E_{nb} \tag{3.6}$$

Where E_f represents the final energy function, which can be divided into two other equations, which denote the energy terms concerning bonded atoms (E_b) (Equation 3.7) and non-bonded atoms (E_{nb}) (Equation 3.8).

$$E_b = E_{bl} + E_{ba} + E_{pta} + E_{ta} \tag{3.7}$$

Where E_b denotes the combination of the bonded energy terms, encompassing bond lengths (E_{bl}), bond angles (E_{ba}), prohibited torsion angles (E_{pta}) and torsion angle values (E_{ta}), respectively.

$$E_{nb} = E_{vdw} + E_{elet} + E_{hb} + E_{solv} \tag{3.8}$$

Where E_{nb} represents the combination of the non-bonded energy terms, comprising the *van der* Waals attraction and repulsion forces (E_{vdw}), electrostatic interactions (E_{elet}), hydrogen bonds (E_{hb}) and implicit solvation components (E_{solv}), respectively.

In this work, although the optimization of structural solutions is performed through changes in the dihedral angles of the target protein structure, the energy evaluations of these models are performed using the Cartesian representation, through the centroid-based energy function of Rosetta[3] (ROHL et al., 2004; KAUFMANN et al., 2010).

[3]<www.rosettacommons.org>

3.7.1 Rosetta Energy Function

The Rosetta energy function (ROHL et al., 2004), implemented by the PyRosetta molecular modeling suite[4] (CHAUDHURY; LYSKOV; GRAY, 2010), was used as an objective function responsible for evaluating the predicted solutions' quality. According to Combs et al. (COMBS et al., 2013), the Rosetta energy function is a knowledge-based energy function developed through an empirical analysis of experimentally determined protein structures of the PDB. It considers biological information, such as radius of gyration (RG), packing density, hydrogen bonding distance, and aa pairwise interactions. Such information is converted into energy terms by Bayesian statistics. It should be noted that the Rosetta energy function is one of the most popular potential functions concerning the PSP field (MOULT et al., 2018; ABRIATA et al., 2018).

Rosetta provides two distinct representations for the atom modeling: centroid-based and all-atom models (LEAVER-FAY et al., 2011). The difference between both consists in the side chain representation, where the first model generates a reduced description, through a higher-level approach, of the 3-D protein structure. So in the centroid-based model, each aa side chain is represented by a centroid located in the chain's center of mass. The second option provides greater atomic detail, where all atoms in the side chain, including hydrogen atoms, are represented (ROHL et al., 2004). As we are interested in the overall folding of protein models, we adopted the centroid-based representation model for energy evaluations in this work.

Commonly, a potential energy function, as mentioned earlier, can incorporate two distinct categories of energy terms (HANDL; LOVELL; KNOWLES, 2008; MACKERREL, 2010): bonded and non-bonded terms. The Rosetta function considers more than 18 energy terms, most derived from knowledge-based terms (ROHL et al., 2004). The function has terms based on Newtonian physics, such as 6-12 Lennard-Jones potential, divided into attraction and repulsion terms needed to describe the *van der* Waals interactions (KUHLMAN; BAKER, 2000), and the Lazaridis-Karplus solvation energy model (LAZARIDIS; KARPLUS, 2000). It also combines terms of knowledge-based interatomic electrostatic interactions obtained through potentials between pairs of amino acids and hydrogen bonding energies dependent on the orientation (KORTEMME; MOROZOV; BAKER, 2003). The scoring function still uses terms to estimate the free energy of conformation-dependent amino acids. These terms aim to evaluate the positioning

[4]<www.pyrosetta.org>

of the aa side chain, according to the Dunbrack rotamer library (DUNBRACK; COHEN, 1997), and validate the conformational preference of the ϕ and ψ angles of the aa, through a Ramachandran plot (RAMACHANDRAN; RAMAKRISHNAN; SASISEKHARAN, 1963). The final energy value of the Rosetta function ($E_{rosetta}$) is given by the summation of all weights performed on the energy terms considered in the calculation. The weight of each term was assigned based on the energy function *Score3*, which is the standard Rosetta function used to evaluate centroid-based structural models (LEAVER-FAY et al., 2013; O'MEARA et al., 2015).

3.7.2 Protein Contact Maps

Predicting protein contact maps (CM) is based on the knowledge discovery from experimental protein structures data and tries to determine which residues are in contact probabilistically. There are several proposed contact map predictors in the literature (SCHAARSCHMIDT et al., 2018). Most of them explore machine learning strategies, such as Deep Learning networks and Support Vector Machines with classical biological features, like SS, solvent accessibility, and sequence profile (ADHIKARI; HOU; CHENG, 2018). Incorporating contact predictions from coevolution-based methods as additional features also significantly improved their performance (SCHAARSCHMIDT et al., 2018; ADHIKARI; HOU; CHENG, 2018).

In the last years, contact predictions were shown to be a valuable addition to the PSP methods (SCHAARSCHMIDT et al., 2018). As reported, improved contact methods can lead to improved FM model accuracy (ABRIATA et al., 2018). However, despite the improvement in the residue-residue contact prediction, its use in an efficient way into the PSP algorithms configures the major challenge (SCHAARSCHMIDT et al., 2018). Various factors determine the methods' performance, such as the number of contacts considered and how they are incorporated in the modeling. Hence, the most suitable contact prediction technique and the number of contacts to consider are dependent on the PSP algorithm.

As pointed out by the Critical Assessment of Protein Structure Prediction (CASP) report (SCHAARSCHMIDT et al., 2018; SHRESTHA et al., 2019; XU; WANG, 2019; KRYSHTAFOVYCH et al., 2019), the use of size lists of $L/2$ contacts can improve the performance, reducing the false positives and taking into account the predicted residue contacts with higher probabilities of being in contact. L represents the target aa sequence length. By the prediction results carried out in the experiments, list sizes of $L/2$ seemed to be one of the best choices. In this work, we used a reduced list of $L/2$ predicted contacts. The CMs were predicted by the TripletRes predictor[5] (LI et al., 2020). TripletRes was ranked as one of the top servers in the 13-th CASP experiment for automated protein contact-map prediction.

In a CM, two amino acids are close enough or in contact if the distance between their Cβ side chain atoms, or Cα of backbone for Glycine, is less than or equal to a distance threshold, generally 8Å. A given term of distance constraint is generally used to get the information from CMs and to overcome some inaccuracies of the energy function (KIM et al., 2014). In this work, besides the Rosetta energy terms, we used a scheme to employ the information of CMs in the problem as a new term in the energy function. This term was idealized based on an atom distance constraint function presented in the work of Kim et al. (KIM et al., 2014) and adapted by Corrêa and Dorn (CORRÊA; DORN, 2019). The CM term is a function of the distances between the aa contained in the CMs. It aims to positively reinforce the aa pairs that are within the contact bounds or to penalize the ones that are out of the threshold, according to Equation 3.9.

$$CM_{term} = \sum_{i,j}^{CM_{pairsL/2}} = \begin{cases} p \times -c, & d(i,j) \leq ub \\ p \times -c \div 2, & ub < d(i,j) \leq ub + 2 \\ p \times +c, & d(i,j) > ub + 2 \end{cases} \tag{3.9}$$

Where p denotes the probability that the residues are in contact, c is a constant, ub is a residue contact upper bound, and $d(i,j)$ represents the Euclidean distance between a pair of amino acids in the predicted contact list. The TripletRes considers the ub contact threshold of 8Å, so we adopted the same threshold of distance. For the constant c, we adopted $c = 1000$ to follow the reinforcement values defined in the SS term, described in the next section.

Thus, for a target protein, the procedure goes through the $L/2$ aa pairs in the predicted CM, measuring the distances between these pairs regarding a given protein model. It gives: (*i*) a positive reinforcement to the term summation, adding a negative constant $(-c)$ multiplied by the probability of the residues are being in contact, if the distance between them is less than or equal to the ub threshold; (*ii*) a positive reinforcement to the term summation but considering the negative constant divided by 2 $(-c \div 2)$, if the distance between the amino acids is greater than the ub but does not exceed $ub + 2$ (tolerance

[5]<https://zhanglab.ccmb.med.umich.edu/TripletRes/>

threshold); or (*iii*) a negative reinforcement to the term summation, adding a positive constant $(+c)$ multiplied by the probability of the residues are being in contact, if the distance between the residues is greater than the threshold $ub + 2$.

3.7.3 Final Objective Function

In addition to the Rosetta and CM energy terms, a SS term (Equation3.10) was incorporated in the objective function to favor the correct formation of secondary structures. The procedure consists of giving a positive reinforcement by adding a negative constant $(-const)$ to the sum of all amino acids of the protein P, when the SS (zp_i) corresponding to the i-th amino acid (aa_i) is equal to the SS (ze_i), equivalent to the same residue but provided as input to the algorithm. On the other hand, the technique negatively reinforces the summation by adding a positive constant $(+const)$, when the SS of the corresponding amino acids are not equal. All amino acids of the protein are compared during the evaluation of the solution. The DSSP (KABSCH; SANDER, 1983) algorithm was used to assign the SS throughout the simulation.

$$SS_{term} = \sum_{aa \in P}^{i+1} V_{aa,zp,ze}(aa_i, zp_i, ze_i) \tag{3.10}$$

$$V_{aa,zp,ze}(aa, zp, ze) = \begin{cases} \textit{-const}, & zp = ze \\ \textit{+const}, & zp \neq ze \end{cases} \tag{3.11}$$

Finally, the terms described above are then added to the Rosetta energy function's result, forming the final evaluation function (E_{final}) adopted in this work (Equation 3.12). It was firstly proposed by Corrêa et al. (CORRÊA et al., 2016) and improved by Corrêa and Dorn (CORRÊA; DORN, 2019).

$$E_{final} = E_{rosetta} + SS_{term} + CM_{term} \tag{3.12}$$

3.8 Final Remarks

This chapter presented concepts of biological foundations required to the understanding of the PSP problem, as well as issues related to it.

The main concepts described include: (*i*) physicochemical protein composition; (*ii*) levels of structural protein abstraction; (*iii*) structural protein classes, describing how the proteins are classified in this thesis; (*iv*) computational representation of protein structures, which defined the adopted computational representation of this work as the set of amino acid dihedral angles; (*v*) PSP problem definition regarding protein stereochemistry, computational modeling, and theoretical formalization of the PSP as an optimization problem; and (*vi*) objective functions for the problem.

4 RELATED WORKS

4.1 Introduction

This chapter presents a review of the related works of global and multimodal continuous optimization through metaheuristic algorithms. Its goal is to focus on the literature concerning search strategies to deal with the optimization issues previously discussed, such as multimodal search strategies, population diversity and convergence, Expr/Expt trade-off, hybrid and parameter control algorithms. The chapter also describes an overview of methods applied to the PSP problem.

Over the last decades, several algorithms have been proposed to solve optimization problems (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). Notably, many of these problems are classified as NP-hard (COOK, 1983), particularly those of real relevance, which means that there is no polynomial-time algorithm or deterministic method to solve them optimally. Usually, these problems are solved with stochastic metaheuristics that allow finding approximate solutions within a reasonable computational cost, but without guaranteeing optimality (TALBI, 2009). Metaheuristics are algorithmic frameworks used to solve a wide range of optimization problems without deeply adapting to each problem. The Greek prefix "meta" is used to contrast with heuristics dependent on the application domain (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). Thus, metaheuristics are good options for problems which there no problem-specific algorithms (KVASOV; MUKHAMETZHANOV, 2018).

According to Boussaïd et al. (BOUSSAÏD; LEPAGNOT; SIARRY, 2013), almost all metaheuristics share similar characteristics, which include: (*i*) bio-inspired concepts, usually based on some principles of biology, physics or ethology; (*ii*) stochastic algorithms, which explore the concept of randomness or random variables; (*iii*) non-dependency of the gradient or Hessian matrix of the objective function; and (*iv*) several parameters that need to be adjusted regarding the problem's domain. Metaheuristics can be classified according to many criteria, in terms of the adopted search strategies and application domain, the use of memory, the technique of neighborhood exploration, or the number of current solutions carried from one iteration to the next (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). A fundamental distinction in literature is the differentiation between single-solution and population-based algorithms. Single-solution metaheuristics, also known as trajectory or LS strategies, start with a single initial solution and move away from it, describing a trajectory in the search space. Typical examples of LS algorithms include the Simulated Annealing (SA), the Solis and Wets (SW), the Tabu Search, the GRASP method, the Variable Neighborhood Search, among others (LUKE, 2013; DOKEROGLU et al., 2019). On the contrary, population-based metaheuristics deal with a population of solutions. The most studied population-based methods are related to bio-inspired concepts of EAs (BACK, 1996) and SI (KENNEDY et al., 2001; LUKE, 2013).

Metaheuristics can deal with many different applications with diverse requirements. This is possible by combining search components regarding the exploration and refinement of the search space, escape from local optima, and determination when satisfactory solutions have been found (TORRES-JIMÉNEZ; PAVÓN, 2014). Nevertheless, particularly in real-life problems, including the PSP, the simple application of canonical methods is not always enough to achieve good performance. This is due to the search space's complexities or the high dimensionality of objective variables (BELDA et al., 2007; HANDL; LOVELL; KNOWLES, 2008). Therefore, several techniques have been considered to improve the metaheuristic's performance. By combining constructive methods with local and population-based strategies, novel hybrid search mechanisms provide more robust and efficient ways to address hard optimization problems (SERGEYEV; KVASOV; MUKHAMETZHANOV, 2018).

4.2 Bio-inspired Metaheuristics

Bio-inspired optimization is an area of artificial intelligence (AI) that has been largely explored over the last decades. Several methods have been proposed, illustrating the application of distinct bio-inspired behaviors and characteristics to handle a near-optimal performance over a wide range of complex problems, such as multimodal optimization functions (SER et al., 2019). The most notable contribution to the field is mainly the analysis, adaptation, and improvement of different search algorithms (SER et al., 2019).

The most known methods are related to concepts of EAs (BACK, 1996) and SI (KENNEDY et al., 2001). EAs are inspired by Darwin's evolutionary theory, where a population of solutions is modified through recombination and mutation search operators. SI algorithms are based on the exploration of simple analogs of social interaction, mimicking the behavioral patterns of biological agents such as fish, birds, and bees (BOUS-

SAÏD; LEPAGNOT; SIARRY, 2013). Typical examples of EAs are the Evolution Strategies (ES), the Genetic Algorithms (GA), the Differential Evolution (DE), and other EA variants, such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and the SHADE algorithm (LUKE, 2013; DOKEROGLU et al., 2019). Besides that, SI, which is a branch of bio-inspired computation that originated from EAs, is based on the emergence of collective intelligence from populations of individuals with simple behaviors for interaction and evolution. Typical methods encompass the Particle Swarm Optimization (PSO), the Artificial Bee Colony (ABC), and the Ant Colony Optimization (ACO) algorithms (SER et al., 2019; DOKEROGLU et al., 2019).

In this thesis work, our research is toward the PSP problem, which comprises a multimodal objective function characterized by the highly rough landscape (HANDL; LOVELL; KNOWLES, 2008). Hence, we intend to deal with multimodal continuous optimization to overcome obstacles imposed by the functions' multimodality over adaptations in the algorithms. Within this context, our concerns are regarding the discovery of optimal or suboptimal solutions of the search space through the development of multimodal strategies capable of keeping a diverse population throughout the process. As stated earlier (Section 2.7), a proper ratio between Expr/Expt can improve the metaheuristic's performance, and the population diversity is strictly related to this in a kind of cause and effect relationship (ČREPINŠEK; LIU; MERNIK, 2013). However, the metaheuristic's search components and parameterization are key factors to control such measures and determine the final optimization results (ALETI; MOSER, 2016; LACERDA et al., 2021).

Thus, regarding the search mechanisms, adaptive and hybrid approaches are commonly explored. The parameter setting is directly related to the metaheuristic's performance. The correct regulation of such values by parameter control strategies requires using the most promising setting over the process to achieve acceptable performance (ALETI; MOSER, 2016; PARPINELLI et al., 2019). In addition, the concept of hybridization in optimization includes the combination of multiple specialized search strategies in discovery and refinement solutions, rules for solution selection, niching-based algorithms focused on diversity preservation, and adaptive parameterization during the algorithm's run (XU; ZHANG, 2014). This concept is based on the principle that a hybrid method by the search mechanism's synergy, such as MAs (MOSCATO; COTTA, 2019), may reach better results than a single strategy to a given problem (SER et al., 2019; ČREPINŠEK; LIU; MERNIK, 2013). The issues regarding multimodal optimization are complementary to each other. Hence the proposed methods may be more broadly designed.

4.3 Parameter Control Strategies

The success of a metaheuristic is strongly related to the correct adjustment of the parameter set that dictates its optimization behavior, such as the choice of representation, parent selection, recombination, and mutation operators (ALETI; MOSER, 2016; PARPINELLI et al., 2019). Manual or off-line parameter adjustment can be challenging and time-consuming as parameters are generally problem-dependent and different values may be optimal at different stages of the optimization process. The parameter search space can become computationally infeasible even for a few options depending on the number of parameters and their explored range of values. Then, automating this task has been one of the most significant challenges in the metaheuristic research field (SER et al., 2019). The automatic parameter adjustment approaches can be divided into two groups as following (HUANG; LI; YAO, 2019; LACERDA et al., 2021; PHAN et al., 2020).

The first group is the parameter or off-line tuning, where satisfactory parameter values are identified before the execution of the algorithm, usually through many previous runs that aim to maximize the chances of success to the target problem (LACERDA et al., 2021). The defined parameter setting remains constant during the whole execution of the optimization process. The parameter tuning can be divided into brute force experiments, experimental design, racing procedures, and meta-optimization (PHAN et al., 2020).

The second group is parameter control or online tuning, where the values of controlled parameters are adjusted according to some strategies during the optimization execution. Usually, in these strategies, the initial values for the algorithm are determined so that it runs for a given amount of time and, then, return a performance measure (LACERDA et al., 2021). The parameter control can identify appropriate values during different optimization stages and gather fitness landscape information to enhance the late stages of the search process. These control strategies can be deterministic, adaptive, or self-adaptive (PARPINELLI et al., 2019; PHAN et al., 2020). Since parameter tuning algorithms define the parameter values before the run and keep them constant during the entire process, only control methods can perform such a dynamic adjustment. So we highlight that in this work, we are particularly interested in parameter control strategies.

Deterministic or predefined algorithms adapt parameters following deterministic rules based on time without relying on any feedback from the search process. Adaptive methods regulate parameters according to feedback from the optimization process as it runs, such as the quality of solutions. The change in algorithm properties serves as guidance to regulate parameter values in the subsequent generations. Self-adaptive methods usually encode the metaheuristics' parameters alongside the objective variables in a single vector, which are evolved together during the process. Better solutions tend to survive to the next generation and propagate their variables along with the parameter set responsible for high-quality (ALETI; MOSER, 2016). Parameters can be grouped according to the adjusted functionality, i.e., representation of individuals, population size, mutation rate/step-size, crossover rate, among others.

According to Lacerda et al. (LACERDA et al., 2021), parameter control methods should regulate any parameter of a metaheuristic, but in most of the works, only part of them are regulated. The others are generally kept constant, defined by tuning methods or in an ad-hoc way. Even the partial parameter regulation makes the problem easier to deal with, simplifying the parameter space. It is noted that the parameters of any metaheuristic do not exert the same influence on the optimization process, which gives them different levels of importance. Thus, it is promising to focus on the most critical ones.

Popular metaheuristics, such as GA, PSO, DE, and ABC, have been largely explored in terms of parameter adaptation in the literature (LACERDA et al., 2021). Parameter control algorithms try to find optimal parameter values for a given period during the optimization. Hence, parameters of search algorithms are regulated through their behaviors in terms of assumed values and optimization state information (ALETI; MOSER, 2016; LACERDA et al., 2021).

Formally, the parameter set P_{set} to be adjusted can be defined as $P_{set} = [\upsilon_1, \cdots, \upsilon_n]$, where n is the number of parameters. Each parameter $\upsilon_i = [\upsilon_{i1}, \cdots, \upsilon_{im}]$ has m possible values, where m is the number of discrete values or the upper bound of a continuous range. The control algorithm aims to find the next parameter value υ_{ij} such that υ_i could positively influence on the method's performance.

The main steps for parameter control include (ALETI; MOSER, 2016): (*i*) feedback, which represents properties (behavior) of a search algorithm in a given optimization stage. The interpretation of the quality of the parameter values depends on the information provided by the feedback strategy, which can encompass the solutions' positions and their fitness values, best fitness, fitness mean or standard deviation, population diversity, and feasibility feedback; (*ii*) parameter effect assessment, which establishes the influence (positive or negative) of parameter values on the algorithm's performance based on the output from the feedback strategy, such as the fitness difference of offspring compared to its parents and the best solution; (*iii*) quality attribution, which tries to determine a successful parameter value for the following iterations. The quality is calculated based on predefined rules that use the effect measured in the previous iterations; and (*iv*) parameter update to the next generation, which is based on the trade-off between using parameter values with high quality and exploring new values. It can update a probability vector for parameter values based on estimated quality and determines how frequently a parameter value is chosen.

Regarding feedback information, most control algorithms use the fitness improvement of solutions to indicate the optimization performance. Also, the population diversity is another feedback information employed as an indicator of the exploration degree of the search space (GINLEY et al., 2011; LIU et al., 2013; ALETI; MOSER, 2016). Besides, the most used strategies for estimating the effect of parameter values on the algorithm's performance are related to the change in the properties of the current solution directly or concerning the best solution, parents, or the entire population (ALETI; MOSER, 2016). For example, in Srinivas and Patnaik (SRINIVAS; PATNAIK, 1994), the effect of mutation and crossover rates is estimated based on the population's best fitness and average fitness. Eiben et al. (EIBEN; MARCHIORI; VALKO, 2004) control the population size based on the fitness improvement of the best solution in the population. The population size grows if the best fitness increases or does not change for a certain number of iterations. The resizing of the population size is focused on big population size when exploration is required and small population size for exploiting the search space.

Parameter quality refers to a scalar value attributed to a parameter to describe its performance based on current or past observations. The most common control methods use a model to predict the parameter quality and the immediate effect as quality. However, the dependency between search components and parameters represents an issue in parameter control. For instance, the Predictive Quality Attribution strategy (ALETI et al., 2014) uses time-series forecasting to infer the performance of parameter values based on past performance. Aleti et al. (ALETI et al., 2014) proposed a parameter control method that uses historical performance measures to approximate the probability distribution that determines the chances of each parameter value producing an optimal result for each generation. Lastly, parameter update strategies mainly encompass greedy schemes, where the parameter value with the best quality is selected, and probability matching techniques, where the probability of applying a specific parameter value is proportional to the quality of that parameter among all possibilities (ALETI; MOSER, 2016).

By relating the topics concerning the metaheuristics' concepts, it is known that the balance between Expr/Expt is an important factor in the success of these algorithms (BOUSSAÏD; LEPAGNOT; SIARRY, 2013; SER et al., 2019). Many of them use this behavior to improve the search performance by controlling parameters as they execute (PHAN et al., 2020). For instance, exploration can be promoted at the beginning of the run by a high level of randomness of solutions' movement. It may also be enhanced during the process when there is stagnation in the search results. Contrarily, the exploitation can be improved by shrinking the search space or lowering the level of randomness. Besides, strategies to control the transition between Expr/Expt phases can be employed. Commonly, they rely on deterministic factors, such as the number of iterations or fitness evaluations of the search algorithm (PHAN et al., 2020). Therefore, most parameter control algorithms are proposed to ensure the proper levels of population diversity in an attempt to balance the Expr/Expt capabilities of the algorithms during the optimization process.

4.4 Hybrid and Memetic Algorithms

Ensemble or hybrid optimization methods are related to using multiple search strategies, subpopulations, algorithms, rules for selection and population replacement, operators, and parameter values to deal with an optimization problem. The idea is that the hybridization of components can reach better results than ensemble composites working on their own. Ensembles of search strategies and components can be classified according to the characteristics of the adopted technique. A low-level ensemble is a method that combines different types of search strategies and operators. Contrarily, high-level ensembles refer to methods that adaptively select the best optimization algorithm for a problem among a set of candidate algorithms (SER et al., 2019).

Thus, one of the main challenges under research related to hybrid methods refers to the specific selection of algorithms to be assembled. Generally, it is possible to choose among a large set of low-level methods or heuristics, but the appropriate selection of these components remains an issue concerning real applications. Also, combining multiple-methods approaches with multiple strategies can lead to challenging problems related to the ensemble's tuning and computational complexity (SER et al., 2019).

One of the most prominent metaheuristics to solve complex optimization problems are the MAs (MOSCATO, 1989; MOSCATO; COTTA, 2019). MAs were proposed as a branch of bio-inspired computation characterizing a specific kind of hybrid evolutionary metaheuristics. It was initially defined as modifications of GAs employing LS mechanisms (SER et al., 2019). Thus, they can be defined as hybrid metaheuristics that incorporate concepts and operators of evolutionary population-based methods for global searches, combined with a simpler LS heuristic responsible for solution exploitation (MOSCATO, 1989). These algorithms are based on the combination of existing algorithmic structures, thus avoiding the limited use of a single method for the problem and providing greater flexibility in dealing with the concerned complexities (MOSCATO; COTTA, 2010; NERI; COTTA; MOSCATO, 2012).

It should be noted that many evolutionary and SI algorithms can perform both global and LS since their search procedure is based on the difference between any pair of individuals, such as in DE and ABC. Therefore, the concept behind MAs has evolved to a more generic approach, defined as the combination of bio-inspired algorithms for global optimization with separate local improvement and individual learning mechanisms, possibly incorporating domain-specific knowledge of the problem at hand (SER et al., 2019). Thus, novel research contributions for MAs have been focused on discovering new synergistic memetic approaches with biological inspiration at their core.

One of the most significant challenges in structuring the MAs is the definition of how the solution space may be explored. It is important to find the correct balancing between global and local search efforts associated with the correct parameterization to achieve good results added to satisfactory performance (MOSCATO; COTTA, 2010; BOUSSAÏD; LEPAGNOT; SIARRY, 2013).

Therefore, a reasonable balance between Expr/Expt is needed to enhance the workability of the method (MOSCATO; COTTA, 2019). For instance, too much exploitation and too little exploration can lead to premature convergence. At the same time, too much exploration and too little exploitation can slow down the convergence and increase the computational costs dramatically. This is uniquely critical in bio-inspired MAs, where diversification is performed by the bio-inspired metaheuristic, and intensification is done by the LS algorithm. Different mechanisms of collaboration between these two search procedures may provide distinct ratios of Expr/Expt. Hence, according to Del Ser et al. (SER et al., 2019), further research is needed towards achieving new procedures for detecting the level of diversity or convergence, as well as suitable countermeasures (diversity or convergence induction) to be inserted in the memetic frameworks (SER et al., 2019). Also, strategies related to adaptive mechanisms to balance such efforts can be investigated.

4.5 Single Global Metaheuristics

Research on single objective continuous optimization, also known as global continuous optimization, represents the foundation for more complex scenarios, such as niching algorithms in multimodal optimization and both multi-objective and constrained optimization algorithms (SER et al., 2019). Traditionally, single objective benchmark problems are the first test for new evolutionary and swarm algorithms (PRICE et al., 2019). For instance, one can transform single global problems into dynamic, niching composition, computationally expensive and other classes of problems. Thus, in the following sections, we describe some relevant metaheuristic works focused on global continuous optimization, then multimodal optimization, considering the previously discussed research topics. Such works address aspects intrinsically related to metaheuristics' performance, such as Expr/Expt abilities, regulation of the population diversity and convergence over the optimization process, hybridization of search algorithms, and parameter control. Furthermore, we highlight that the development of this thesis is based on the investigation of these optimization concepts aiming at providing an optimizer for search problems regarding the multimodal continuous domain.

In this work, we intend to present an adaptive MA framework with a LS strategy for multimodal optimization, combined with a modified ABC algorithm used as an exploratory method to support the MA. In addition, the modified ABC was implemented as a hybrid algorithm that uses a self-adaptive strategy for its control parameters, population division based on the solutions' quality, and DE-based mutation operation to enhance the search abilities of the algorithm. According to the literature, we believe that the synergy of strategies incorporated in the framework seems suitable to the multimodal continuous domain in terms of search space exploration and refinement of solutions.

Thereby, the methods presented in this work were designed based on a constructive approach, encompassing a more general MA focused on single global continuous optimization for multimodal functions, a MA to deal with multimodal problems with many global optima, and a latter variant of the MA for the real problem of 3-D PSP. The following sections present relevant works related to the algorithms implemented in this work, such as variants of the ABC, DE algorithm, multimodal and niching algorithms, and methods for the PSP problem.

4.5.1 Artificial Bee Colony Algorithm

The ABC consists of a metaheuristic based on the foraging behavior of a bee colony, which was recently proposed, focusing on multivariate numerical function optimization (KARABOGA; BASTURK, 2007; AKAY; KARABOGA, 2012). ABC has become a popular optimizer because it presents few control parameters, simple structure, ease of implementation, and robust optimization performance (GAO; LIU; HUANG, 2012; DOKEROGLU et al., 2019). Several studies have been published demonstrating its competitiveness on a large set of benchmark functions when compared with other population-based metaheuristics, such as GA, PSO, and DE algorithms (KARABOGA; BASTURK, 2008; KARABOGA; AKAY, 2009; KARABOGA et al., 2014).

In a real bee swarm, some tasks are performed by specialized individuals whose aim is to maximize the amount of nectar stored in the hive through division of labor and self-organization (AKAY; KARABOGA, 2012). The foraging task is crucial to a honeybee, and it depends primarily on the ability of individuals to be recruited and to abandon depleted food sources. The ABC simulates three types of existing bees, responsible for the foraging task (KARABOGA; BASTURK, 2007): (*i*) employed bees; (*ii*) onlooker bees; and (*iii*) scout bees. Half of the colony includes employed bees, and the other covers onlooker bees. The intelligent behavior that emerges from the cooperation and interaction of bees, necessary for foraging food, is represented in the following stages (AKAY; KARABOGA, 2012).

1. Employed bees' stage: employed bees are responsible for exploiting the food sources they have previously known. Returning to the hive, they store the collected nectar while sharing information with the onlooker bees about the quality of their food sources through the *Waggle* dance in the hive designated area. The nature of the dance is related to the quality of the food source and aims to recruit new bees. Good food sources tend to attract more onlooker bees;
2. Onlooker bees' stage: onlooker bees await in the hive for the return of the employed bees. After the *Waggle* dances, they decide which food sources are more profitable to be exploited based on the information shared by the employed bees;
3. Scout bees' stage: when an employed bee realizes that its power supply is depleted, it abandons the food source and becomes a scout bee. Scout bees search randomly over the search space for new food sources.

Thus, the bees designated to carry out the food foraging can provide sustenance for the swarm from the interaction between these three stages. Thus, employed and onlooker bees are responsible for exploring the already known food sources, while scout bees must randomness search around the environment for new places of exploitation.

Regarding the standard ABC (Algorithm 1), each food source represents a problem solution, and its quality is expressed by its fitness value. Each food source is exploited by only one employed bee, i.e., the number of employed bees corresponds to the same number of food sources (number of population solutions). The number of onlooker bees is equal to the number of employed bees. Let SN be the number of food sources, eb be the number of employed bees, and ob be the number of onlooker bees, then $SN = eb = ob$. As previously described, ABC simulates three stages of the bees' foraging process. The modeling in computational terms is given as follows:

1. Each population solution represents a food source, which is updated in this step through a mutation search equation. Each employed bee is associated with only one food source, and there are no solutions without employed bees;
2. ob solutions of the population are generally selected by roulette wheel or ranking-based selection strategies, simulating the behavior of the onlooker bees, and the same updating process of the previous step is applied to the selected solutions;
3. The most inactive population solution is discarded, and a new one is generated. The most inactive solution is the one that does not suffer improvements for a given number of generations, regarding a "limit" parameter l.

The update operation used in steps 1 and 2 creates a new food source around an already known one. Thus, the generation of a new solution $\upsilon_i = [\upsilon_{i1}, \upsilon_{i2}, \cdots, \upsilon_{iD}]$ from the i-th existing solution $X_i = [x_{i1}, x_{i2}, \cdots, x_{iD}]$, such that υ_i replaces X_i ($X_i = \upsilon_i$) if it is better, is given by Equation 4.1.

$$\upsilon_{ij} = x_{ij} + \delta_{ij}(x_{ij} - x_{kj}) \tag{4.1}$$

Where $i = 1, \cdots, SN$ and $j = 1, \cdots, D$. SN denotes the number of solutions in the population and D represents the problem dimension. x_{ij} represents the j-th objective variable of X_i, υ_{ij} represents the new value assigned to x_{ij} and x_{kj} represents the j-th variable of the k-th population solution ($k = 1, \cdots, SN$), randomly selected among all solutions of the population, such that $k \neq i$. δ_{ij} is the mutation scaling factor, which is a random real number in the range $[-1, 1]$. In each mutation operation for X_i, only a single variable j, randomly selected, is mutated. The update ends with a greedy selection between the new solution v_i and the former X_i regarding their fitness function values. The Algorithm 1 shows the pseudocode of the standard ABC. It receives as input parameter the population pop to be optimized, the scout bee threshold l, and executes until the defined stop criterion is satisfied. The stop criterion can be determined by a given number of generations to be executed.

Algorithm 1 Pseudocode of the standard ABC algorithm.

```
Require: $pop$: population to be optimized; $l$: threshold of discard; stop criterion
Ensure: $pop$: population of optimized solutions
 1: $NS = eb = ob \leftarrow$ number of solutions in $pop$
 2: $D \leftarrow$ problem dimension
 3: while stop criterion is not satisfied do
        //Employed bees' stage
 4:     for each $pop_i$ in $pop$, $i \leftarrow 1 : eb$ do
 5:         $j \leftarrow rand(1, D)$
 6:         generate a new value for $v_{ij}$ from $pop_{ij}$ applying the Equation (4.1)
 7:         check whether the new value is within the dimension range of variable $j$
 8:         replace $pop_i$ for $v_i$, if $v_i$ is better than the original $pop_i$
 9:     end for
10:     $SortPopulation(pop)$ from best to worst according to the fitness values
        //Onlooker bees' stage
11:     for $i \leftarrow 1 : ob$ do
12:         $sol \leftarrow$ probabilistic selection of solution from the $pop$
13:         $j \leftarrow rand(1, D)$
14:         generate a new value for $v$ from $sol$ applying the Equation (4.1)
15:         check whether the new value is with in the dimension range of variable $j$
16:         replace $sol$ for $v$, if $v$ is better than the original $sol$
17:     end for
18:     $SortPopulation(pop)$ from best to worst according to the fitness values
        //Scout bees' stage
19:     for each $pop_i$, $i \leftarrow 1 : SN$ do
20:         check that the solution $pop_i$ has not received improvements than $l$ generations
21:         discard $pop_i$
22:         insert a new solution in the population
23:     end for
24:     $SortPopulation(pop)$ from best to worst according to the fitness values
25: end while
26: return $pop$
```

Therefore, the performance of the ABC is based on the search strategies and control parameters. The control parameter components are the population size SN, scaling factor δ and the “limit” parameter l.

4.5.2 Relevant Artificial Bee Colony Variants

It is remarkable that the Expr/Expt of solutions are extremely important mechanisms in the ABC. However, the algorithm presents some inefficiencies, like other metaheuristics, such as being good at search space exploration but not in the refinement of solutions, which can lead to a slow convergence rate (AKAY; KARABOGA, 2012; LI; NIU; XIAO, 2012). Thus, some ABC variations have been proposed over the years to find an ideal balance between these two processes and accelerate the algorithm's convergence. These modified versions can be roughly divided into two categories (CAI et al., 2020): (*i*) modification made in the solution search equation; and (*ii*) hybridization with other search strategies. According to the reported results, these modified versions can perform better than the original ABC (ZHU; KWONG, 2010; AKAY; KARABOGA, 2012; LI; NIU; XIAO, 2012).

One of the most known improved variants is the gbest-guided ABC algorithm (GABC), proposed by Zhu and Kwong (ZHU; KWONG, 2010). According to the authors, even though the canonical ABC has been proved to be a good option on many engineering problems, it still converges slowly because of its poor exploitation ability. This version was proposed as an alternative to enhance the exploitation ability of the ABC. In GABC, the information of the global best (gbest) solution is used in the solution search equation to guide the search towards the best individual and hence find more accurate solutions than the standard ABC. The gbest search equation of GABC is given by Equation 4.2 and modifies the Equation 4.1.

$$v_{ij} = x_{ij} + \delta_{ij}(x_{ij} - x_{kj}) + \gamma_{ij}(gbest_j - x_{ij}) \tag{4.2}$$

Where $gbest_j$ is the j-th variable of the global best solution found so far, and γ_{ij} is a random real number in the range $[0, 1.5]$. However, according to Gao et al. (GAO; LIU; HUANG, 2013), it should be noted that Equation 4.2 may cause an oscillation phenomenon, when the guidance of the last two terms may be on opposite directions regarding X_i, and hence could degrade the convergence (CUI et al., 2016; GAO et al., 2018).

In another ABC version, Akay and Karaboga (AKAY; KARABOGA, 2012) proposed modifications in the components that control the frequency of mutation operation (rate-MR). They also presented an analysis of the most suitable parameterization to be used in the ABC, such as the scaling factor that determines the magnitude of change in parameters while generating a new solution and the “limit” parameter that represents the discard threshold for scout bees. As already mentioned, in the standard ABC, only one objective variable of a given solution is updated at each mutation operation, implying slow convergence for the algorithm. Therefore, in the modification rate-MR, each variable j of the solution X_i is updated according to the designed control parameter MR. Updating more than one variable can enhance the exploitation and hence accelerate convergence. Thus, a variable is updated with a probability of $MR\%$, if $rand(0, 1) < MR$, where $rand$ generates a real number in the range $[-1, 1]$. The lower value for MR may imply slow improvement of solutions, while higher values for MR can cause an unnecessary diversification in the population. By suggestions of the authors, $MR = 0.4$ or close to it is a good setting. Also, the authors suggested $l \approx 200$ or the commonly proportion $l = SN \times D$ for the “limit” parameter in continuous optimization as good options. Where SN is the population size and D represents the problem dimension. It is notable that the authors of both works concluded that ABC could be considered a good deal in terms of global and local optimization due to the different tasks simulated by the bees.

In the work of Cao et al. (CAO et al., 2019), an improved ABC (IABC) was proposed to enhance the performance of the already mentioned ABC variant GABC. In the IABC, the employed bees use the same search equation of the GABC, but the onlooker bees employ a modified solution search equation based on the gbest solution to improve further the exploitation. In the GABC, both employed and onlooker bees generate new solutions by changing only one dimension of their parent solutions according to the same search equation, as in the standard ABC. So the modified search equation for onlooker bees considers more dimensions to be mutated with $MR = 0.3$, as presented in Akay and Karaboga (AKAY; KARABOGA, 2012), increasing the influence of the best solution on the equation.

Kiran and Findik (KIRAN; FINDIK, 2015) proposed a directed ABC algorithm with control parameter (rate-MR) to accelerate the convergence of the method. The solution search equation in the original ABC is entirely random in terms of direction because the scaling factor δ_{ij} is a random real number between $[-1, 1]$ (Equation 4.1). This undirected search can slow the algorithm’s convergence, such as the oscillation phenomenon, previously mentioned. Thus, the authors incorporated direction information for each variable of solutions in the population. The proposed strategy modifies the original search equation (Equation 4.1) in the generation of v_i as follows:

$$v_{ij} = \begin{cases} x_{ij} + \delta_{ij}(x_{ij} - x_{kj}) & \text{if } (d_{ij} = 0) \\ x_{ij} + r(|x_{ij} - x_{kj}|) & \text{if } (d_{ij} = 1) \\ x_{ij} - r(|x_{ij} - x_{kj}|) & \text{if } (d_{ij} = -1) \end{cases} \tag{4.3}$$

Where d_{ij} is the direction for the j-th variable of the i-th solution, δ_{ij} is a random real number in range $[-1, 1]$ and r is a random real number in range $[0, 1]$. The Equation 4.3 identifies the search direction considering the previous value of variable j. In the initialization of the algorithm, the direction for all variables is equal to 0 ($d_{ij} = 0$). Then, if v_i is better than X_i and the previous value of the j-th variable is less than the current value, the direction of this variable is updated to -1 ($d_{ij} = -1$); otherwise the direction is set to 1 ($d_{ij} = 1$). If v_i is worse than X_i, the direction of the variable is set to 0 ($d_{ij} = 0$). Besides, the method also incorporates the modification rate-MR with $MR = 0.3$, proposed by Akay and Karaboga (AKAY; KARABOGA, 2012) and described above, to update more objective variables than one.

The work of Gao et al. (GAO et al., 2015) presented an ABC algorithm based on information learning of the search space (ILABC). As observed above, the ABC search equation (Equation 4.1) does well in exploration but not in exploitation. According to the authors, the coefficient δ_{ij}, which is a random number, is enough for exploration. Nevertheless, the equation's direction vector is also chosen randomly, not considering the population neighborhood to guide the search. Hence, using this information, such as in the multi-population technique, can help the algorithm enhance the balance between Expr/Expt of the neighborhood.

Therefore, at each generation of the ILABC, a clustering partition divides the entire population into multiple subpopulations with size M, regarding the Euclidean distance. With this, different subpopulations focus on searching different subregions, and the ABC is applied to each one. We note that this niching-based strategy is further detailed in the following section, concerning the multimodal metaheuristics (Section 4.6). The size of each cluster is dynamically adjusted based on the previous search experience. In the beginning, the algorithm is randomly assigned a M value from set S, e.g., $S = \{5, 10, 25\}$. At each iteration, the size M changes based on the quality of the global best solution. If the fitness value of the global best solution is not improved after one iteration, it means that the current M is not suitable for the search. Thus, a new M is randomly generated.

However, population clustering can lead to an ineffective approximation of the optimal solution, whereas different subpopulations work independently without communication among them. In single global optimization, it is necessary to consider the influence of the gbest solution to accelerate the convergence, which is different from multimodal optimization that aims to locate more than one global optimum. Thus, two search mechanisms were designed to promote the exchange of information in each subpopulation and between different subpopulations. It incorporates in the solution search equation for employed and onlooker bees the information of the local best solution of each cluster and the gbest solution of the entire population, respectively.

Cui et al. (CUI et al., 2016) proposed a depth-first search (DFS) framework to better balance the Expr/Expt in ABC. The DFS aims to allocate more computing resources to the best solutions throughout the optimization and enhance the exploitation mechanism. In the employed bee phase of the DFS, only some randomly selected solutions are mutated. Thus, if a better solution v_i is found in the neighborhood of a randomly selected solution X_i, X_i is replaced by v_i, and then it is updated consecutively until a worse one is produced. Similarly, in the onlooker bee phase, only the top T elite solutions are randomly selected to exploitation, where $T = p \times SN, p \in (0, 1)$. Thus, a randomly selected elite food source is exploited to generate a solution until a better one cannot be produced. The number of onlooker bees is $r \times T$, where $r \in 1, \cdots, ceil(1/p)$. The authors also introduced two solution search equations inspired by DE variants. Such equations lie in the fact that they present no bias, as the oscillation phenomenon, to any search direction. The first one incorporates the information of elite solutions applied to the employed bees. The second equation also exploits the information of the gbest and elite solutions simultaneously in the onlooker bee phase. The combination of these mechanisms yields the DFSABC_elite algorithm.

The work of Li et al. (LI et al., 2017) introduced a gene recombination operator (GRO) into the algorithm to accelerate its convergence. The method was inspired by the natural phenomenon that good individuals also present good genes, and the combination of superior genes could more easily produce better offspring. In GRO, only a part of good solutions in the population, which may contain satisfactory information in different dimensions, are selected to generate new solutions by recombination. Hence, the solutions could converge to the global optimum from different directions, and the recombination between the better individuals may ease the approach to the global optimal solution. The best $n = ceil(q \times SN)$ solutions in the population are treated as elite solutions, where

$q \in (0, 1]$ and SN is the population size. The authors adopted $q = 0.1$ and $SN = 50$. GRO chooses probabilistically two solutions from the elite group to recombination, considering their distances. The two selected parents may have a significant difference in some dimensions. Such procedure is applied at the end of each generation of the ABC.

In Anam (ANAM, 2017), the author proposed a hybridization between the ABC and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (ABC-BFGS) for continuous optimization. BFGS serves as an iterative LS for finding local optima. In the proposed method, after its execution, the final best solution is then submitted to the BFGS to be further exploited.

Cui et al. (CUI et al., 2017) presented an ABC variant with an adaptive method for the population size (AMPS), called APABC. The algorithm aims to achieve a better balance between Expr/Expt by adaptively controlling the population size. AMPS has based on the natural concept that the availability of food resources influences the size of a population. Then, when the algorithm performs well in exploration, AMPS shrinks the population to promote exploitation by periodically removing low-quality solutions and storing them into an external archive. On the contrary, if the algorithm performs well in exploitation, AMPS enlarges the population to improve exploration by introducing randomly selected solutions from the archive to keep the population diversity. With this, the population size adaptively adjusts according to the evolution process status. Besides, to keep the diversity when the population size becomes small, the authors proposed a random search equation inspired by the DE mutation operator.

In Gao et al. (GAO et al., 2018), the authors proposed two new updating equations (ILTD_ABC) for ABC to enhance the algorithm performance. The first one employs an intelligent learning strategy that uses the gbest solution influence and borrows from the PSO algorithm its fast convergence characteristic, by including the fitness values of the entire population in the search equation for employed bees, except for the worst solution. In this search equation, the fitness values and positions of the entire population and the current gbest position cooperate to determine the step size of the newly employed bees. The worst employed bee is updated by a distinct search equation that uses information from the best individual in the previous and current iterations to accelerate its convergence. Different from the other employed bees, the worst updates all the objective variables during each iteration. The controlling parameters of the mutation equation are generated by a turbulent distribution to balance Expr/Expt efforts. To select onlooker bees to mutation, a threshold that selects individuals to form an elite group containing better solutions is proposed. If the bee satisfies this threshold, it is selected as a group member with better performance. All the solutions from this elite group are updated considering the influence of some other solution from this elite group.

In Xue et al. (XUE et al., 2018), a self-adaptive ABC algorithm based on the gbest solution (SABC-GB) for global optimization was proposed. According to Xue et al., the search algorithms frequently implement only one candidate solution generating strategy (CSGS), which tends to be either good at exploration or good at exploitation. Thus, SABC-GB employs distinct CSGSs simultaneously. This modification aims the balance adjustment between the convergence rate and the algorithm's exploration. The method keeps a pool of search equations that are used adaptively to generate new individuals according to their previous performance (window of 10 generations) in generating promising solutions. Each CSGS is selected probabilistically through the roulette wheel selection and is updated based on the total success and failure ratio. The adaptive pool is composed of three solution search equations, which are the gbest equation, presented earlier in Equation 4.2, the ABC/best/1, and ABC/best/2, proposed by Gao et al. (GAO; LIU; HUANG, 2012). Besides, SABC-GB adopts chaotic systems (ALATAS, 2010) and opposition-based learning (RAHNAMAYAN; TIZHOOSH; SALAMA, 2008) in the initialization phase to avoid local minima.

Cai et al. (CAI et al., 2020) proposed a dynamic best neighbor-guided search strategy (DNABC) for the ABC. In the proposed strategy, a dynamic neighborhood of the parent individual with variable size is constructed concerning different evolution stages of the algorithm. The neighborhood size is small at the beginning of the process, but its size gradually increases as the optimization proceeds. The construction of the neighborhood is based on the individual index, where M solutions are randomly selected from the entire population as the neighbors of X_i. Then, the best solution from the neighborhood is selected to guide the search to achieve a better balance between Expr/Expt. This algorithm is an improved version of the best neighbor-guided ABC (NABC) (PENG; DENG; WU, 2019), which does not use a dynamic construction in the neighborhood strategy. Besides, the work presented an improved global neighborhood search operator, emphasizing the randomly selected solutions to escape from local optima. Such an operator searches the neighborhood using small steps as a LS technique. The operator aims to increase the chance of finding a better solution, considering that local optima usually have a relatively good fitness value and a better solution may exist within or near the individual's neighborhood.

The work of Jadon et al. (JADON et al., 2017) presented a hybridization of ABC and DE algorithms to provide a more efficient method than ABC and DE. Although ABC and DE are two very popular metaheuristics, both algorithms may present unbalanced Expr/Expt and slow convergence rates. Then, the authors suggested that the hybridization of ABC and DE should provide a method with a better ratio between Expt/Expt mechanisms and convergence speed than using a single metaheuristic. In the algorithm, named Hybrid Artificial Bee Colony with Differential Evolution (HABCDE), the employed bee phase was modified by employing the influence of the gbest solution. The onlooker bee phase was inspired by DE search equations, while scout bees were also modified for higher exploration.

In Chen et al. (CHEN; TIANFIELD; LI, 2019), the authors presented the self-adaptive differential ABC (sdABC) algorithm. The sdABC incorporates three distinct search strategies from the DE algorithm in both employed and onlooker bee updating phases. Employing differential search strategies, more variables are modified each time based on the combination of mutation and crossover, and the crossover rate control can also adjust the number of variables being updated parameter. Besides, the selection of search strategies is calculated using a probability-based self-adaptive mechanism, which can select the most appropriate search strategies for the algorithm depending on the optimization stage.

Table 4.1 summarizes the main components and differences among ABC methods presented above.

4.5.3 Differential Evolution Algorithm

As mentioned before, several metaheuristics have been applied with success to solve hard optimization problems (BOUSSAÏD; LEPAGNOT; SIARRY, 2013; SER et al., 2019). Currently, one of the most remarkable EA is the DE algorithm (STORN; PRICE, 1997). The DE is a stochastic algorithm for solving numerical continuous optimization problems. Since its advent, the algorithm has become a powerful global optimizer (DAS; SUGANTHAN, 2010; DAS; MULLICK; SUGANTHAN, 2016; ELTAEIB; MAHMOOD, 2018).

The DE is a population-based algorithm that has a population of NP individuals. Each individual $X_i = [x_{i1}, x_{i2}, \cdots, x_{iD}]$ is a vector of D variables, which is the problem dimension. A randomly initialized population is evolved, guiding solutions in a search

Table 4.1: Summarization of the ABC variants and their main components

Method	Modified Equation	gbest	Adaptive	Hybridization
GABC	X	X		
IABC	X	X		
Directed ABC	X			
ILABC	X	X	X	Neighborhood strategy
DFSABC_elite	X	X		DFS; DE search equation
GRO_ABC				Genetic recombination operator
ABC-BFGS				BFGS
APABC	X	X	X	AMPS; DE search equation
ILTD_ABC	X	X	X	
SABC-GB	X	X	X	Initialization strategy
NABC	X	X		Neighborhood strategy
DNABC	X	X	X	Neighborhood strategy
HABCDE	X	X		DE search equation
sdABC	X		X	DE search equation

Source: From the author (2022).

space toward a global optimum. At the end of the process, i.e., after a given number of generations or fitness evaluations, the algorithm returns the best-fitted individual as the final solution. Over each generation, DE uses three operations for each individual, i.e., mutation, crossover, and selection (NERI; TIRRONEN, 2010; DAS; MULLICK; SUGANTHAN, 2016).

Similar to the update operation in ABC, a new solution v_i is created using one of the mutation search equations from the i-th existing solution X_i. The “DE/rand/1” strategy is one of the most adopted in DE. This operation randomly selects two individuals and their difference multiplied by a scale factor $F \in [0, 1]$ is added to the third randomly selected individual. “DE/rand/1” is given as follows:

$$v_i = X_{r1} + F(X_{r2} - X_{r3}) \tag{4.4}$$

Where $i = 1, \cdots, NP$. The indexes $r1$, $r2$, and $r3$ are randomly chosen within a set $[1, NP]$, such that $r1 \neq r2 \neq r3 \neq i$. DE applies the mutation to all solutions of the population.

After mutation, an individual ν_i is generated by a binomial crossover as follows:

$$\nu_{ij} = \begin{cases} \upsilon_{ij} & \text{if } rand(0,1) \leq C_r \text{ or } j = j_{rand} \\ x_{ij} & \text{otherwise} \end{cases} \tag{4.5}$$

Where $i = 1, \cdots, NP$ and $j = 1, \cdots, D$. $C_r \in [0, 1]$ is the crossover rate and represents the probability of using the variables for ν_i from the mutant individual υ_i. $j_{rand} \in \{1, \cdots, NP\}$ is a randomly selected index responsible for ensure that ν_i contains at least one variable from υ_i. After crossover, the selection compares X_i and its corresponding trial candidate ν_i according to their fitness function values by a greedy scheme. Thus, the performance of the algorithm is based on the search components and the control parameters. The control parameter components are the population size NP, scaling factor F, and the crossover rate C_r.

Despite the potential of DE, such as in ABC, some adjustments to the standard algorithm are essential to enhance its performance. For instance, stagnation, premature convergence, and sensitivity to control parameters are the main issues that influence the performance of DE (ELTAEIB; MAHMOOD, 2018). Hence, as exemplified above, many improved DE variants and hybrid algorithms have been proposed to solve problems in different domains (DAS; SUGANTHAN, 2010; DAS; MULLICK; SUGANTHAN, 2016).

Therefore, in the work of Brest et al. (BREST; MAUČEC; BOŠKOVIĆ, 2017), a new variant of the DE algorithm, named as jSO, was presented. The algorithm is based on the L-SHADE algorithm (TANABE; FUKUNAGA, 2014), which was the winner of the Congress on Evolutionary Computation (CEC) 2014 competition on single objective real-parameter optimization[1] (LIANG; QU; SUGANTHAN, 2013). The jSO was on the second place on the CEC 2017 competition on single objective real-parameter optimization[2] (AWAD et al., 2016). The competitions for single objective real-parameter optimization are important test benchmarks for high-accuracy computing, where the goal is to find a global optimum on solving complex optimization functions.

The L-SHADE (TANABE; FUKUNAGA, 2014) is an extension of the Success-History based Adaptive DE (SHADE) algorithm (TANABE; FUKUNAGA, 2013) with the Linear Population Size Reduction mechanism (LPSR). L-SHADE uses the success-history-based adaptation from SHADE, which is a mechanism for parameter adaptation based on the historical memory of successful parameter values previously found through-

[1]<https://personal.ntu.edu.sg/EPNSugan/index_files/CEC2014/CEC2014.htm>
[2]<https://personal.ntu.edu.sg/EPNSugan/index_files/CEC2017/CEC2017.htm>

out the process. Such mechanism employs a historical memory with H entries (M_{C_r}, M_F) that stores a set of C_r and F values that have performed well in past generations and generate new C_r, F pairs by sampling the parameter space close to one of these stored values through Gaussian and Cauchy distributions, respectively.

The algorithm employs the "current-to-pBest/1" mutation strategy to generate a new solution from X_i ($i = 1, \cdots, NP$) by using the previously defined parameter values Cr_i and F_i. At the end of each generation, the memory contents (M_{C_r}, M_F) are updated based on the sets of successful parameters S_{C_r}, S_F by the weighted Lehmer mean. The algorithm also adopts an external archive to preserve diversity.

Besides, L-SHADE uses the LPSR, a simple deterministic population resizing procedure that reduces the population linearly as a function of the number of fitness evaluations. LPSR continuously decreases the population to match a linear function where the population size at generation 1 is NP^{init}, and the population at the end of the run is NP^{min}. After each generation g, the population size in the next generation, NP_{g+1}, is given as follows:

$$NP_{g+1} = round\left[\left(\frac{NP^{min} - NP^{init}}{Max\_FEV}\right) \cdot FEV + NP^{init}\right] \tag{4.6}$$

Where NP^{min} is set to the smallest possible value such that the algorithm can be applied, FEV is the current number of fitness evaluations, and Max_FEV is the maximum number of fitness evaluations. Whenever $NP_{g+1} < N_g$, the exceeding worst-ranking individuals are deleted from the population. Therefore, L-SHADE self-adapts scale factor and crossover rate parameters and shrinks population size during the optimization.

As jSO is based on L-SHADE, it presents improvements mainly regarding the parameter values for C_r and F stored in the historical memory (M_{C_r}, M_F), and the "current-to-pBest/1" mutation strategy. Thus, the remaining components in jSO, such as the external archive and linear population size reduction, are the same as in L-SHADE.

In another work, Brest et al. (BREST; MAUČEC; BOŠKOVIĆ, 2019) presented a new version of DE (jDE100) for global continuous optimization. The algorithm is based on the self-adaptive jDE algorithm (BREST et al., 2006), which is a popular version of DE, previously proposed by the same authors. The jDE is among the state-of-the-art algorithms for global optimization (DAS; SUGANTHAN, 2010; DAS; MULLICK; SUGANTHAN, 2016). The jDE100 algorithm uses two populations in an attempt to maintain an efficient exploration throughout the optimization. It employs a self-adapting mechanism in the scaling factor and crossover rate parameters, where each individual has its own control parameter values F_i and Cr_i. Besides, it uses a simple one-way migration of the currently best individual among the populations to promote convergence. However, the algorithm performs restart mechanisms separately in both populations to manage population diversity.

It should be noted that jDE100 was the winner of the 100-Digit Challenge on single objective real-parameter optimization[3] (PRICE et al., 2018), considering the results from the CEC 2019, the Genetic and Evolutionary Computation Conference (GECCO) 2019 and the Swarm, Evolutionary and Memetic Computing Conference (SEMCCO) 2019 (PRICE et al., 2019). Such results represent the potential of DE algorithms in facing complex optimization problems. In the next section, some strategies are presented to deal with multimodal optimization problems, and many of the existing methods are DE-based algorithms.

4.6 Multimodal Metaheuristics

Multimodal metaheuristics are primarily designed to find all possible optimal points and not just a single solution to the problem (DAS et al., 2011). If an optimization problem requires more than one global optimum, it can be considered a multimodal optimization problem. The task of locating distinct optima in a single run makes it more complicated than locating just a single global optimal solution (QU; SUGANTHAN; LIANG, 2012). Traditional population-based metaheuristics are primarily designed for single global optimization, then modifications and multimodality-specific mechanisms are needed to locate multiple optima simultaneously (WANG et al., 2017). However, multimodal strategies frequently appear in literature also as a reference to single optimization on a multimodal fitness landscape. Also, multimodal or niching algorithms have first appeared in research on population diversity preservation within metaheuristic algorithms (LI et al., 2016). Considering that a multimodal algorithm searches for multiple optima in parallel, the probability of getting trapped on a local optimum may be reduced. Thus, the success of bio-inspired algorithms in real-world applications has also been accompanied by their uses of niching methods (LI, 2009).

Multimodal optimization may increase the probability of a metaheuristic finding global optima since search efforts are not concentrated just in one region of the search

[3]<https://github.com/P-N-Suganthan/CEC2019>

space but in different areas. The discovery of solutions in distinct regions of the search space can also maintain a diverse population, preventing a premature convergence to local optima, and hence improving the metaheuristic performance (GLIBOVETS; GULAYEVA, 2013; LI et al., 2016; SER et al., 2019). However, it can be observed that evolutionary metaheuristics tend to naturally converge to a single global optimum due to the genetic drift inherent to the population evolution (BELDA et al., 2007). With this, the preservation of population diversity by maintaining multiple solutions throughout the algorithm's execution configures one of the main challenges concerning metaheuristics over multimodal optimization.

As mentioned in Section 2.3.1, the most common strategies used in multimodal optimization are based on the niching concept (MAHFOUD, 1995; GLIBOVETS; GULAYEVA, 2013), which concerns the attempt to find and preserve multiple solutions around distinct niches of the search space. The concept of niching is inspired by the way organisms evolve in nature. In the case of single multimodal optimization, they may prevent premature convergence better populating the fitness landscape (LI et al., 2016). Thus, multimodal optimization can be achieved using a niching strategy, incorporated into a global optimization metaheuristic, which can be called the core algorithm, to enable parallel convergence to distinct optima (AHRARI; DEB; PREUSS, 2017b). However, it is highlighted that simply preserving a high level of population diversity is not enough for niching since a high population diversity could encompass only random points. This means that a niching algorithm must converge locally to global optima, which induces the formation of distinct subpopulations around promising regions of the search space (neighborhood of optimal solutions) (QING et al., 2008; GLIBOVETS; GULAYEVA, 2013). Regardless of the niching strategy adopted, it is necessary to find global optima by efficiently exploring the search space and refinement of discovered regions and preserving them throughout the optimization process (SER et al., 2019).

Specifically for multimodal optimization with multiple global optima, the effective identification of optimal points, besides the maintenance of distinct solutions, is not trivial (WANG et al., 2019). According to Li et al. (LI et al., 2016), the metaheuristic's population does not have to totally converge to single solutions, each corresponding to a global optimum. For instance, one can store the identified global optima into an external archive, separate from the current optimization population (EPITROPAKIS; LI; BURKE, 2013; LACROIX; MOLINA; HERRERA, 2016). Another approach is employing a large population of solutions, where the population could find some equilibrium state. Some solutions would keep oscillating around a stable optimum in an equilibrium state, but without reaching complete convergence (LI, 2009).

Another issue concerning niching algorithms is the need to designate niche parameters. As their optimal values depend on the problem at hand and may vary during the evolution, setting them properly is a hard task (WANG et al., 2019). The most representative parameter for niching is the niche radius, which needs to be specified to indicate how far apart the optima are from each other (LI et al., 2016).

4.6.1 Classical Niching Algorithms

One of the most known niching strategies is the Crowding algorithm and its variants (JONG, 1975; THOMSEN, 2004). Firstly, the crowding concept aims to eliminate the most similar solution when a new one enters a subpopulation. In the former Crowding method (JONG, 1975) only a fraction of the entire population is selected for search operations in each generation. An offspring generated from randomly selected parents is compared to this subpopulation, and the most similar individual is replaced by a greedy scheme (GLIBOVETS; GULAYEVA, 2013). However, the algorithm commonly needs a crowding factor (CF) parameter to determine the size of the subpopulation. Thus, in Mahfoud (MAHFOUD, 1992), the Deterministic Crowding (DC) algorithm was proposed to eliminate the CF parameter. The DC randomly selects two parents for crossover and mutation from the entire population. The two generated offspring are compared with their parents, where they replace the nearest parent by a greedy scheme.

The crowding strategy can be combined with many other search algorithms (DAS et al., 2011). For example, in Thomsen (THOMSEN, 2004), crowding was used in the DE algorithm. In this algorithm, when the canonical DE generates an offspring, it competes only with the most similar individual of the entire population, measured by Euclidean distance. The offspring replaces its parent also by a greedy competition. In this variation, the CF is equal to the entire population.

Similar to the Crowding algorithm, the Restricted Tournament Selection (RTS) defines which solutions are replaced in the population to insert new ones (HARIK, 1995). For each offspring generated by search operators and solutions from the whole population, the algorithm defines a random sample of “w” (window size) solutions and determines which one is the nearest to the offspring by Euclidean distance. The nearest individual within the sample competes with the offspring by greed. Such a strategy avoids too dissimilar solutions from competing with each other but needs the specification of the window size parameter (DAS et al., 2011).

The Sharing algorithm (GOLDBERG; RICHARDSON et al., 1987) attempts to deal directly with the locations and preservation of multiple solutions. It aims to divide the population into different niches according to the similarity of solutions (DAS et al., 2011). The algorithm modifies the search space by reducing the fitness values in densely populated regions. It decreases each solution's fitness in relation to the number of similar individuals in the population. The algorithm needs a niche radius to determine whether solutions are in the same group.

Another commonly used niching strategy is the Speciation, based on the species conservation concept (LI et al., 2002). It intends to separate the population into several species according to their similarities (LI et al., 2002; DAS et al., 2011). Different niches are formed around a dominating solution called the species seed. Solutions are then assigned to the niche with the closest species seed to them. The search operations are carried out within each species. However, the definition of a species involves a parameter of species distance (species radius). The parameter defines the upper bound on the distance between two solutions for which they are similar.

To mitigate the effects of the niching methods from the sensitivity to parameters, some less parameter-sensitive niching strategies were proposed (YANG et al., 2016b). For instance, the Hill-Valley (HV) niching test (URSEM, 2000; MAREE et al., 2018) samples enough equidistantly located intermediate test points within the line segment connected by two solutions to detect HVs. If there exists at least one point whose fitness is worse than those of both individuals, then a valley is detected, indicating these two individuals belong to different niches. Moreover, the recursive middling strategy (YAO; KHARMA; GROGONO, 2009) was proposed to reduce the number of sampled points in the HV. Like a binary search, it continuously samples the middle point of the line segment connected by two updated endpoints until the demanded point is found or the two endpoints converge to the same one. Although these methods are promising in niching the population, they usually cost a large number of fitness evaluations to detect all valleys. Besides that, clustering algorithm integrated into niching strategies (GAO; YEN; LIU, 2013) can be used to tackle such parameter issues. In this case, cluster-based niching methods transfer the sensitive parameter (niche radius) to a less sensitive parameter (cluster size).

Nevertheless, according to Li et al. (LI et al., 2016), commonly, the removal of the niche radius parameter concerning these classical niching algorithms inevitably introduces new parameters. Therefore, to reduce such parameter dependency, several niching variants have been proposed (LI, 2009; QU; SUGANTHAN; DAS, 2013; EPITROPAKIS; LI; BURKE, 2013; LI; TANG, 2014). Among them, some of the most popular are the neighborhood-based methods, which attempt to avoid specifying the niche radius (WANG et al., 2019). Such an idea aims to partition a single population into multiple subpopulations regarding a specific neighborhood metric. Each subpopulation represents a local subspace of the search space and covers a small number of optimal solutions. The search process is then performed separately within these subpopulations. However, in this case, the difficulty is how to define the area for each region and how to generate subpopulations (GAO; YEN; LIU, 2013).

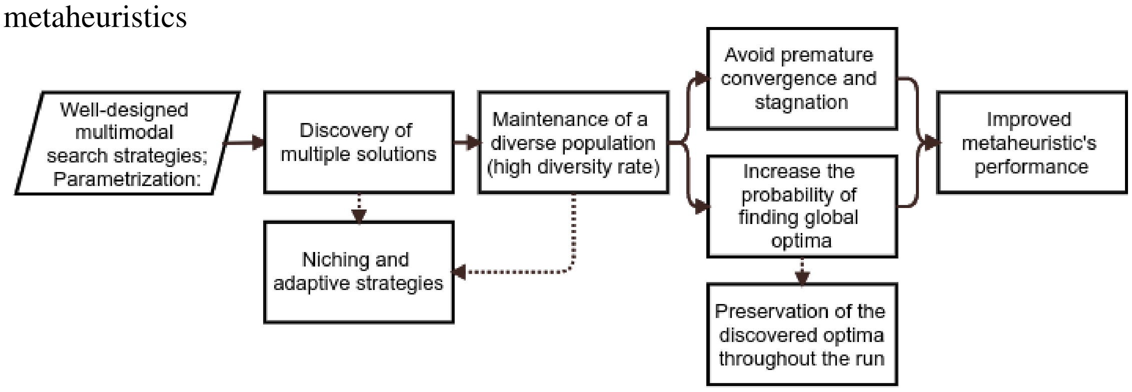
In this sense, the neighborhood representation can be either index-based or distance-based (SER et al., 2019). The index-based topology forms a neighborhood of an individual by using adjacent indices in the population and a predefined topology (e.g., tree, ring, or star) over the search space (LI, 2009). The distance-based niching forms the neighborhood relationship between individuals in the population by using their adjacent Euclidean distances (QU; SUGANTHAN; LIANG, 2012; WANG et al., 2017).

Also, adaptive parameter control has attracted considerable attention to deal with the excessive parameterization concerning multimodal methods. If properly designed, an adaptive strategy can enhance the performance of a metaheuristic by dynamically adapting the parameters to the characteristics of different fitness landscapes. The convergence rate can be improved if the control parameters are adapted to suitable values at different evolution stages of a specific problem (GAO; YEN; LIU, 2013). Figure 4.1 shows the concerns summarized above regarding multimodal optimization by niching metaheuristics. In the following, we present some remarkable works concerning the aspects discussed in this section.

4.6.2 Relevant Niching Algorithms

The work of Li (LI, 2009) described a *lbest* PSO niching algorithm using a ring neighborhood topology, which does not require any niching parameters. The paper aims to overcome the need for niching parameter specification, which is typically difficult to set as they are problem-dependent. In a lbest PSO, the position of each particle is influenced only by the best-fit solution chosen from its neighborhood. In a ring topology, the particles' neighborhood encompasses only its immediate neighbors on its left and right

Figure 4.1: Summarization of the concerns regarding multimodal optimization by niching metaheuristics


Well-designed multimodal search strategies; Parametrization:
Discovery of multiple solutions
Maintenance of a diverse population (high diversity rate)
Avoid premature convergence and stagnation
Increase the probability of finding global optima
Improved metaheuristic's performance
Niching and adaptive strategies
Preservation of the discovered optima throughout the run

Source: From the author (2022).

according to the individual indices. A PSO algorithm using the ring topology can assume niching characteristics by using individual particles' local memories to form a stable network retaining the best positions found so far, while these particles explore the search space more broadly. Instead of using a single global best, each particle is attracted toward a fitter local best (best-fit personal best) only within its immediate vicinity. As the process evolves, multiple niches are formed around optima in parallel. Based on this idea, niches are formed naturally without the need to specify any parameter. The ring topology naturally divides the population into multiple subpopulations, each operating as a separate PSO with its own local neighborhood best. Given a reasonably large population uniformly distributed in the search space, the ring topology can provide stable niches across different local neighborhoods, eventually locating multiple global optima.

In Qu et al. (QU; SUGANTHAN; DAS, 2013), a distance-based locally informed particle swarm (LIPS) algorithm was presented. This method avoids the need to specify any niching parameter and enhances the searchability of PSO. According to the authors, most of the existing PSO-based niching algorithms are difficult to use in practice because of their poor LS ability and requirement of prior knowledge to specify niching parameters. Thus, instead of using the global best particle, LIPS uses several local bests to guide the search of each particle. Besides using its personal best, the algorithm adopts the local information from its nearest neighbor, which is estimated in terms of Euclidean distance. The position and velocity of each swarm's particle is updated by a novel equation that considers its neighborhood. The neighborhood of a given solution is formed by the $nsize$ nearest particles. The neighborhood size $nsize$ parameter is dynamically increased from 2 to 5 over the fitness evaluations. LIPS can operate as a stable niching algorithm with distinct niches that may converge to different global peaks.

Gao et al. (GAO; YEN; LIU, 2013) presented a cluster-based DE with a self-adaptive strategy for multimodal problems. The method adopts a multi-population scheme, where the entire population is divided into subpopulations based on spatial positions of the individuals by clustering. The clustering procedure is a distance-based neighborhood algorithm. It aims to partition the population into multiple disjoint subpopulations, each comprising M solutions with adjacent locations in the search space. A given solution is selected from the population as a seed to form a niche by including $M - 1$ individuals closest to the seed. All the niches are formed according to this process. After the population is completely divided, each niche is then evolved independently based on DE mechanisms.

The clustering is incorporated in two common niching DE algorithms, known as crowding DE (CDE) (THOMSEN, 2004) and species-based DE (SDE) (LI, 2005), which yield self-CCDE and self-CSDE, respectively. Both methods transfer the sensitive parameter, i.e., the CF or the species radius, to the cluster size, which is a less sensitive parameter. According to the authors, a multi-population scheme addresses the population diversity, whereas a self-adaptive parameter control strategy guides the convergence to the optima contained by different subpopulations. The algorithms employ parameter control strategies in DE, regarding the scaling factor parameter F of the mutation equation and the crossover rate parameter C_r of the crossover operator (ZHANG; SANDERSON, 2009). Due to the control parameters adapted to appropriate values based on different evolution stages, the methods can improve the convergence rate and accuracy.

The work of Qu et al. (QU; SUGANTHAN; LIANG, 2012) presented a neighborhood mutation strategy integrated with implementations of niching DE algorithms, namely neighborhood-based CDE (NCDE), SDE (NSDE), and neighborhood-based sharing DE (NShDE). The authors argued that many niching methods are integrated with DE to make it suitable for multimodal optimization. However, a few works focus on DE mutation operation, which is critical for efficiently solving multimodal optimization problems. In the proposed neighborhood-based metaheuristic, the mutation operation is performed within each Euclidean neighborhood. The method limits the mutation within some distance-based neighborhood solutions, thereby effectively locating multiple optima. The neighborhood concept allows higher exploitation of the regions piloting the mutation moves. In neighborhood mutation, difference vector generation is limited to a number M of similar individuals (neighborhood size) measured by Euclidean distance.

Each solution is evolved toward its nearest optimal point. This parameter controls how many individuals are selected in each subpopulation. According to the authors, M should be chosen between $1/20$ and $1/5$ of the population size. It can also be dynamically set, from a relatively large value to a small value. Different from other niching parameters, neighborhood size is easy to choose as it can be made proportional to the population size. The authors have shown that varying the parameter value within this range does not interfere with algorithms' performance. Also, the authors presented a comparative survey on niching algorithms and their applications.

In Wang et al. (WANG et al., 2017), the authors pointed out three key issues of the existing multimodal algorithms: (*i*) the dilemma to adopt search operators that favor both Expr/Expt. The dilemma consists that algorithms not only require high diversity for exploring multiple global optima but also need convergence to refine the solutions in each globally optimal region; (*ii*) concerning the selection operator to form the next generation population, where the existing methods do not consider selecting solutions regarding different peaks. The selection procedure is commonly performed in terms of the metaheuristic's mechanisms, such as the replacement by nearest parent or elitism; and (*iii*) most algorithms are not aware of the convergence regarding different search areas, leading to an unnecessary optimization in these spaces.

Then, to address these drawbacks, a dual-strategy DE (DSDE) with affinity propagation clustering (APC) (FREY; DUECK, 2007) based selection and archive technique was proposed (WANG et al., 2017). The work adopted the same clustering species-based DE algorithm described above in Gao et al. (GAO; YEN; LIU, 2013). This clustering procedure needs the specification of the parameter cluster size M. Thus, the authors also used the dynamic cluster size niching (YANG et al., 2016a) to vary the parameter M adaptively. The interval for M is set as $[4, 20]$ because DE must have at least four individuals. The niche size is chosen randomly in every generation. This adaptation tries to prevent the premature convergence of niches, increasing the cluster size, or enhance the exploitation capability, decreasing the cluster size. Inside the cluster-based DE, a dual-strategy mutation scheme is used, enabling each solution to choose its suitable mutation strategy and balance Expr/Expt search requirements of its own. After the clustering process, each subpopulation is divided into two equal parts according to the solutions' fitness. The first part encompasses individuals with better fitness values for exploitation, whereas the second one has suitable individuals for exploration. For the selection issue, an adaptive selection mechanism based on APC is proposed to choose proper solutions from different optimal regions for locating different peaks in the landscape. At the end of each generation, without replacing the generated offspring, the APC divides the entire population (parents and offspring) into some non-overlapping groups, where each one should converge to one or a small number of optima. The selection of solutions to form the next generation population is probabilistically performed based on the fitness of the species seed regarding each subpopulation. This probabilistic model prioritizes the best individuals in each cluster, keeping the population diversity with more suitable solutions close to global optima. Finally, an archive technique is applied to detect converged individuals. These individuals are stored in the archive to keep promising solutions and are reinitialized to explore new areas. A solution is treated as stagnant if it has not been improved over a certain number T of generations. For each stagnant solution, the M nearest neighbors, based on Euclidean distance, are found in the whole population to obtain a converged subpopulation. Thus, the converged solution itself and those worse than it are reinitialized and stored in the archive. The individuals better than the converged solution are still kept in the population to maintain the exploitation ability.

In another paper from the same authors (WANG et al., 2019), an automatic niching DE (ANDE) based on the APC technique was presented. In this work, APC is used as a parameter-free automatic niching method that does not need to predefine the number of clusters or the cluster size M. In the ANDE algorithm, after the partition of the population into clusters automatically by the APC to locate different optimal regions, the basic DE is performed within each niche. Then, after each generation, a contour prediction approach (CPA) is employed to estimate the landscape contour of each niche.

The CPA is a predictive search strategy, which considers the distribution information of some solutions in the niche to predict the rough position of the potential optima. However, according to the authors, as the potential optimum predicted by CPA is in a rough position, it may still not be accurate enough. Thus, a two-level local search (TLLS) strategy is further performed after the CPA to enhance the exploitation around these regions. Gaussian distribution is adopted due to its promising LS performance by sampling small areas. The TLLS has a two-level local search, including a niching-level LS and an individual-level LS for finding promising solutions. On niching-level, the LS is probabilistically performed on niche seeds (best solution of a niche) based on their objective values by the ranking-based selection scheme. If a niche is selected for LS on niching-level, some individuals with better fitness in this niche should also do LS. Then, an individual-level LS is probabilistically applied to its solutions also by the ranking-based selection. These three main components play different roles in the algorithm and compensate each other.

To deal with a similar dilemma of the Expr/Expt balance of search operators pointed out above (WANG et al., 2017), the work of Hong et al. (HONG et al., 2020) presented a multi-angle hierarchical differential evolution (MaHDE) algorithm designed from two main aspects. Firstly, a fitness hierarchical mutation (FHM) strategy is used to consider the individuals' fitness to match them with distinct mutation equations and balance the Expr/Expt efforts. The FHM divides the individuals into two (low/high-level) based on their fitness values in the current niche. The niching process is made similar to the crowding algorithm, where each individual of the population forms its niche considering the K nearest neighbors, given by Euclidean distance. The best individual of a niche guides the low-level individuals to increase the convergence. In contrast, the high-level solutions are guided by themselves to keep their superiority in terms of fitness. We note that such a strategy is similar to the work described above but adopting mutation equations with distinct purposes. Secondly, a directed global search (DGS) and an elite local search (ELS) are employed in the late evolution stage to increase the population diversity for locating more peaks and improve the accuracy of the solutions found over the optimization, respectively. The DGS strategy is applied to the low-level solutions by expanding their search ranges and thus supporting them to escape from local optima. The ELS strategy is only performed on the high-level solutions, which works via Gauss perturbation for its narrow sampling space. It not only can refine the solution accuracy but also can save the number of fitness evaluations.

Yang et al. (YANG et al., 2016b) presented an adaptive multimodal ACO (AM-ACO) with a niching algorithm, which can preserve high diversity to deal with multimodal continuous optimization. In ACO, each ant constructs solutions using a Gaussian kernel function based on individuals selected probabilistically from an archive. The construction strategy provides high diversity to ACO. However, ACO cannot be directly applied to multimodal problems because the selection and construction operations are based on global information, only suitable for single optimization. According to the authors, there is no previous work on extending ACO to solve multimodal problems. The AM-ACO operates on the niche level by incorporating niching instead of operating on the whole archive as in traditional ACOs. So before ants construct solutions, the already found solutions in the archive are divided into several niches regarding the used niching strategies. The algorithm uses the cluster-based procedure incorporated in two well-known niching strategies, i.e., crowding and speciation. The clustering adopts a random-based niche size setting scheme to tackle the dilemma that niche size is problem-dependent. Specifically, the niche size is randomly selected from a predefined niche size set containing small and large integers during each generation. Hence, the AM-ACO can adapt to the fitness landscape and each subregion of a given problem. Also, the method employs an adaptive adjusting strategy for the σ parameter in the solution construction phase, which takes the differences among niches into consideration. To accelerate convergence, a DE mutation operator is alternatively used to build (shift) base vectors for ants to construct new solutions. Then, as promising solutions are generally found around the best ones, a LS based on Gaussian distribution is adaptively performed around the niche's seeds with probabilities based on their fitness to enhance the exploitation.

In Ahrari et al. (AHRARI; DEB; PREUSS, 2017b; AHRARI; DEB; PREUSS, 2017a), a niching method called the Covariance Matrix Self-adaption Evolution Strategy with Repelling Subpopulations (RS-CMSA) was presented. The algorithm can learn the relative size and possibly the shape of the basins. No assumption on the distribution of global minima is required. In the RS-CMSA, many subpopulations explore the search space in parallel through instances of the core search algorithm CMSA, such that solutions of each subpopulation maintain a distance from a number of taboo points. The taboo points encompass the centers of better subpopulations and previously identified optima, stored in an internal archive for each niche. The distance-based rejection defines taboo regions around the taboo points by the normalized Mahalanobis distance, where a subpopulation may not produce any offspring. The repelling radius of an archived point is adapted based on the number of subpopulations that converge to that basin. After the execution of the CMSA algorithms, the method uses the HV test to verify whether the best member of a terminated subpopulation refers to a new basin. If the new member shares the same basin with an archived solution, it replaces the archived solution only if it is fitter. A strategy to update the taboo regions addresses the challenge of basins of distinct shape, size, and numbers. When all subpopulations converge, a new set of subpopulations with an increased population size are restarted. Thus, the core search algorithms are furthermore kept away from previously located global optima. It is noticed that the performance of RS-CMSA was evaluated on the test functions of the CEC 2013 for multimodal function optimization benchmark (LI; ENGELBRECHT; EPITROPAKIS, 2013), where it was the winner of the GECCO 2017 competition on niching methods for multimodal function optimization[4].

In Yang et al. (YANG et al., 2016a), taking advantage of estimation of distribution algorithms (EDAs) in preserving high diversity, Yang et al. presented a multimodal EDA (MEDA). According to the authors, current EDAs are primarily designed for single optimization, and a few attempts have been reported in literature focused on EDAs for multimodal problems. The algorithm is integrated with clustering strategies for crowding and speciation. The clustering procedure adopts a randomness-based dynamic niche sizing strategy to reduce the sensitivity to the cluster size in the niching algorithms and balance the Expr/Expt efforts. After partitioning the population into niches, MEDA starts to estimate the probability distribution of each niche. Then, taking advantage of Gaussian and Cauchy distributions, the offspring are generated at the niche level alternatively using these two distributions instead of only using Gaussian distribution as in traditional EDAs. The selection of promising individuals is made according to the adopted niching strategies. Further, solution accuracy is enhanced through a LS scheme based on Gaussian distribution. Gaussian distribution is employed because it owns a narrow sampling space, especially when the standard deviation is slight, which is beneficial for local refinement. The LS is probabilistically performed around seeds of niches with probabilities determined adaptively according to the fitness values of the seeds.

The work of Maree et al. (MAREE et al., 2018) presented a Hill-Valley (HV) Clustering, a simple algorithm to adaptively cluster the search space in niches, such that a single optimum resides in each niche. The core of HV Clustering is the HV niching test (URSEM, 2000) combined with the concept of the nearest better tree clustering (PREUSS, 2010; PREUSS, 2012). The HV test can be used to detect whether two solutions belong to the same niche, as previously described.

In the HV Clustering, the best solution of the entire population forms the first cluster. Then, the HV test is used to verify if the second-best solution belongs to the same niche as the first one. If it belongs, the second-best solution is added to the cluster of the best solution. Otherwise, a new cluster is formed. Next, the third-best solution is tested against the nearest solution that has better fitness. If it does not belong to the same niche as its nearest better solution, it is tested against the second nearest better solution. If it also does not belong to that niche, a new cluster is created. For each solution, the $D + 1$ nearest better neighbors are tested, where D is the problem dimension. If a solution does not belong to any of the niches of its nearest neighbors, it forms a new cluster. In each

[4]<http://www.epitropakis.co.uk/gecco2017/>

of the located niches by the HV Clustering, a core search algorithm and a restart scheme are combined to optimize that niche, yielding the Hill-Valley EA (HillVallEA). The HillVallEA is equipped with different core search algorithms, including CMSA (BEYER; SENDHOFF, 2008) and four variants of AMaLGaM (BOSMAN; GRAHL; THIERENS, 2013). These algorithms are based on Gaussian distribution and return a single solution. After running all core search algorithms, a post-processing step is performed to discard the local and duplicated global optima. Then, all remaining optima are tested for being in a different niche by the HV test and added to an elitist archive. Finally, the algorithm is restarted, and all elite solutions are added to the new population. If no new global optima are found in a run, restarts are performed with increased population size to detect smaller niches. We observe that HillVallEA was the winner of the GECCO 2018/2019 competitions on niching methods for multimodal function optimization[5] [6].

In Chen et al. (CHEN et al., 2019), a distributed individuals differential evolution (DIDE) algorithm is proposed based on a distributed individuals for multiple peaks (DIMP) framework. The DIMP framework provides sufficient diversity by letting each individual act as a distributed unit to find a peak, avoiding the issue of population division, and maintaining sufficient diversity to locate more peaks. As each solution is a distributed unit in the framework, the DE mutation may be challenging because it often requires another solution from the same unit. Then, the authors designed a virtual population for each individual generated based on the original individual. The construction of a virtual population is controlled by an adaptive range adjustment (ARA) strategy, where the range of the virtual individuals is initially large and gradually decreases to explore the search space sufficiently for locating a peak and then gradually approach it. The authors also presented two mechanisms integrated with the DIMP, i.e., lifetime and elite learning mechanisms (ELM). The lifetime mechanism is inspired by the natural phenomenon that every organism gradually age and has a limited lifespan. When an individual runs out of its lifetime, it is reinitialized with a new lifetime to increase diversity to locate more optimal points. If a reinitialized solution presents good enough fitness regarding an access criterion, it becomes an elite solution and is stored in an archive. Contrarily, the ELM aims to refine the accuracy of elite solutions in the archive. However, to avoid unnecessary elite learning on elite solutions located at the same optimum, all the elite solutions are clustered. Only the best fitness solution in each cluster is selected for elite learning. Similar to the work of Wang et al. (WANG et al., 2019) described above, the ELM

[5]<http://www.epitropakis.co.uk/gecco2018/>
[6]<http://www.epitropakis.co.uk/gecco2019/>

exploits the area around the selected elite solutions based on the Gaussian distribution. An exponential descent strategy is proposed to adaptively adjust the standard deviation of the Gaussian distribution to sufficiently exploit the solution and improve the solution accuracy.

Zhao et al. (ZHAO; ZHAN; ZHANG, 2020) developed an adaptive guidance-based differential evolution (AGDE) with archive strategy. The AGDE uses the information of the nearest individuals regarding the current solution to form an adaptive mutation strategy (AMS), which makes each one moves towards the nearest peak. It is used to overcome the difficulties of a single mutation operator in guiding the population evolution adaptively. The mutation is divided into two situations according to the problem dimensions. The first case is when the problem dimension D is less than or equal to 3. In this case, the current solution is guided by itself using an original DE mutation. The second scenario is when D is more than 3. This mutation considers the two nearest individuals of the current solution. Then, the best individual among them is used to guide the current solution, whereas the others and more two random individuals help accelerate the population convergence.

The AMS includes a global perturbation based on the solutions' fitness to improve the exploration. The method also encompasses an iterative feedback archive (IFA) strategy that uses an archive to store the global optima in every iteration. Moreover, similar to other described works, a Gaussian disturbance-based elite learning (GDEL) strategy is performed in the archive to refine the solutions' accuracy. These new solutions are used to feedback the population during the optimization to enhance the diversity. The archive is cleared every T iterations to ensure the stored solutions positively impact the evolution. After Gaussian disturbance in the archive, all the solutions are added to the population while the worst individuals are removed. Thus, the AMS strategy helps locate more peaks, while the IFA and GDEL can maintain the found solutions and refine their accuracy.

In Liu et al. (LIU et al., 2021), the authors proposed a Self-adaptive Double-Layer-Clustering Speciation Differential Evolution (SDDLCSDE) algorithm for multimodal optimization. Based on the species-based clustering technique, previously explained, the first layer divides the entire population into independent subpopulations to locate global optima. Then, each niche evolves independently by using the DE algorithm to maintain the population diversity. According to the authors, as in a single global DE, the multi-population strategy also suffers from the diversity loss problem during the later stages of evolution. Thus, unlike other single-layer clustering methods, a global DE search is integrated as the second layer into the algorithm. The species seeds selected from each subpopulation of the first layer form a new niche to detect the missed optima in the search space. It may increase the exploration ability, which helps individuals escape from local optima and prevents the loss of optimal solutions. Also, the algorithm uses a self-adaptive parameter control in DE, concerning the scaling factor parameter F of the mutation and the crossover rate parameter C_r of the crossover operator, similar to those presented in Zhang and Sanderson (ZHANG; SANDERSON, 2009) and Gao et al. (GAO; YEN; LIU, 2013). For each individual, The parameter updates are based on the Cauchy and normal distributions, respectively.

Dubois et al. (DUBOIS; DEHOS; TEYTAUD, 2021) proposed a tree-based niching method that hierarchically divides the search space and estimates the attractive regions using a reinforcement learning technique. According to the authors, numerous metaheuristics are based on restarting evolution strategies, such as the CMA-ES, for multimodal optimization. These algorithms generally perform many local searches for finding all the global optima of the fitness function. However, the strategy used to sample and select the restarting points for LS is a crucial step.

Thus, the proposed algorithm uses an upper confidence tree (UCT) as a niching algorithm. The UCT is a value-based reinforcement learning method, a variant of the Monte Carlo Tree Search (MCTS) (KOCSIS; SZEPESVÁRI, 2006). The policy used in the UCT is based on a multi-armed Bandit algorithm (AUER; CESA-BIANCHI; FISCHER, 2002), designed to solve the exploitation versus exploration problem. A node in the tree corresponds to a search space region, and its children partition this region according to one dimension. The evolution strategy is used to traverse the tree by selecting interesting children recursively. As an evolution strategy for niche optimization, the random restarts with decreasing step-size (RDS) algorithm is adopted, which consists of a single-optimum LS controlled by a restart strategy. The LS used is the (1+1)-ES. The algorithm iteratively computes local searches until all the optima are found, or the maximum number of function evaluations has been reached. Thus, it learns the attractive regions to sample the restarting points for these local searches based on reinforcement learning. The main goal of the method is to select interesting regions while still exploring the whole search space.

The work of Li et al. (LI; YU; TAKAGI, 2019) presented a two-stage niching algorithm that separates local optima areas in the first stage and searches the optimum point of each area using any optimization technique in the second stage. The first stage is based on the Nearest-better Clustering (NBC) algorithm (PREUSS, 2010; PREUSS, 2012) and aims to detect niche areas in which local optima are far from each other in high-dimensional search spaces.

To elucidate, the NBC (PREUSS, 2010; PREUSS, 2012) creates nearest better spanning trees and forms niches from these trees. First, it creates a weighted spanning tree by connecting each solution to the nearest solution with better fitness, measured by the Euclidean distance. Then, the spanning tree is split into several connected subtrees by removing its long edges. Each subtree represents a distinct niche over the search space. The algorithm employs two simple rules to detect and remove long edges: (*i*) all edges that lengths are longer than a threshold are removed, which is given by the average length of all edges; and (*ii*) considering a node (solution) with more than three incoming edge connections, an edge is removed if its length is more than $\beta \times d$, where d is the median of the incoming edges of this node and β is given in terms of the the population size and the problem dimension.

Thus, Li et al. (LI; YU; TAKAGI, 2019) proposed a weighted gradient and distance-based clustering method (WGraD) and two methods for determining its weights to form niches and overcome NBC. The WGraD creates spanning trees (clusters) by connecting each solution to another suitable one decided by weighted gradient information and weighted distance information among solutions, which denote the climbing function between a pair of nodes. Such function is a linear combination of gradient information and distance information. For weight determination, the first method forms a unique spanning tree. Then it uses a dynamic pruning method and the HV test to cut long edges and merge tiny subtrees that belong to the same niche. The second method assigns different weights to different nodes based on distance information. Both methods, WGraD1 and WGraD2, employ the DE algorithm as the metaheuristic for niche optimization after clustering local areas.

In Wang et al. (WANG; ZHAN; ZHANG, 2019), a niching technique based on minimum spanning tree (MST) with DE algorithm (MSTDE) was presented. Similar to the above presented work, the solutions in the search space represent nodes of the tree. In every generation, a complete weighted graph is built based on the distribution of individuals, where each pair of nodes is connected by a weighted edge measured by the Euclidean distance. Then, an MST is built on the weighted graph, and the M largest weighted edges of the MST are pruned to form $M + 1$ subtrees. The DE is executed within each subpopulation, where a distinct solution selection was adopted. When an offspring is generated, it is compared with its nearest neighbor and replaces the neighbor only if it is the better. Besides, a dynamic pruning ratio (DPR) strategy is proposed to randomly determine the parameter M and reduce its sensitivity in the niching algorithm. Moreover, a distributed computational model is employed in MSTDE, where different subpopulations run simultaneously using several virtual machines (VMs).

Wang et al. (WANG et al., 2019) presented a multilevel sampling strategy based on the memetic differential evolution (MMDE) algorithm. The multilevel sampling strategy dynamically divides the whole population into multiple levels according to the fitness of solutions at each generation. Then, a subpopulation is sampled adaptively from the individuals at different levels to undergo a balanced evolution process based on a niching method. The niching applied to the subpopulation consists of the clustering SDE, described above in Gao et al. (GAO; YEN; LIU, 2013). Besides, a crossover-based local search (XLS) is employed to refine the seed solutions of niches during the evolution. The MMDE uses a parameter adaptation mechanism to control the DE parameters. Such scheme maintains a historical memory of H entries for the parameters F and C_r (TANABE; FUKUNAGA, 2013). The parameter updates are based on the historical memory and the Cauchy and normal distributions, respectively.

In the work of Lacroix et al. (LACROIX; MOLINA; HERRERA, 2016), the authors described the Region-based Memetic Algorithm with Archive (RMAwA) for multimodal problems. It uses a region-based niching strategy, which divides the search space into predefined and indexable hypercubes with decreasing size. The algorithm stores the most promising regions in an external archive to maintain the population diversity. Besides, the most promising solutions are improved with a LS algorithm, which is applied until it has reached a local or global optimum. Thus, regions intensively explored by LS are excluded from further exploration with the EA and stored in the indexed archive to reduce the search space. Also, the number of optima that RMAwA can identify is not limited by the population size since the optima found are stored in the archive and not only in the population.

The work of Zhang et al. (ZHANG et al., 2019) described a tree-structured random walking bee swarm optimizer for multimodal landscapes. The authors designed an MST-based niching method, characterized by a tree structure and a random walking process, embedded into the ABC algorithm (TS-ABC). The strategy constructs a complete weighted graph based on the solutions' positions. Then, an MST that encodes the distribution of the solutions is built upon the complete graph to guide the population search, similar to (WANG; ZHAN; ZHANG, 2019). Each candidate solution represents a tree node and moves along the edges to gather information about the search space. The dance trajectories of bees are simulated by random walks on the MST in a probabilistic manner considering distance and fitness information. The solutions are updated by adding differences of the position vectors selected from the dance trajectories. By the probabilistic random walking on the MST, the employed bees and onlooker bees can perform structural searches. Thus, the bees not only perform exploitation within the niches, which are formed by the MST clustering, but they also have some possibilities to walk outside to explore new areas. Moreover, a honeycomb structure used as an archive is introduced to store information of the solutions that have been fully exploited. An employed bee is redirected to other search regions through reinitialization once its food source is exhausted.

Table 4.2 summarizes the main components and differences among the multimodal methods described above.

Table 4.2: Summarization of the discussed multimodal algorithms and their main search components

Method	**Core Algorithm**	**Niching Strategy**	**Adaptive Niching**	**Archive**	**Parameter Control**	**Hybridization**
`lbest PSO`	PSO	Ring-based neighborhood				
`LIPS`	PSO	Distance-based neighborhood	Niche size			
`Self-CCDE/CSDE`	DE	Cluster-based CDE/SDE	Cluster size		In DE	
`NCDE/NSDE/NShDE`	DE	Distance-based neighborhood	Neighborhood size			
`DSDE`	DE	Cluster-based SDE	Cluster size	X	In DE	Dual mutation; APC selection
`ANDE`	DE	APC				CPA; TLLS
`MaHDE`	DE	Crowding				FHM; DGS; ELS
`AM-ACO`	ACO	Clustering crowding/speciation	Cluster size	X	In ACO	DE mutation; Gaussian LS
`RS-CMSA`	CMSA	Taboo regions; HV test	Taboo region size	X		
`MEDA`	EDA	Clustering crowding/speciation	Cluster size		In LS application	Gaussian LS
`HillVallEA`	CMSA; AMaLGaM	HV clustering; NBC	Cluster size			Restarting
`DIDE`	Distributed DE	DIMP framework	ARA strategy	X	In ELM	Lifetime mechanism and ELM
`AGDE`	DE	Distance-based neighborhood		X		AMS; IFA; GDEL
`SDDLCSDE`	DE	Cluster-based SDE			In DE	Global DE search
`Tree-based niching`	RDS algorithm	UCT	Regions location		In RDS	(1+1)-ES
`WGraD`	DE	NBC; HV test	Number of niches			
`MSTDE`	DE	MST-based niching	Number of niches			Distributed niches
`MMDE`	DE	Cluster-based SDE	Cluster size		In DE	XLS
`RMAwA`	MA	Region-based niching	Region size	X		LS
`TS-ABC`	ABC	MST-based niching	Number of niches	X		Random walking

Source: From the author (2022).

4.7 Computational Methods and Metaheuristics Applied to the PSP

Over the last years, knowledge-based protein modeling has consolidated itself as a mature research field, becoming one of the leading research resources in this context. However, despite the significant progress made, the PSP represents an extremely challenging problem, and further research is needed to understand the protein folding mechanisms fully. Such orientations were delineated from results' analysis of the CASP experiments[7], whose objectives are to determine the state-of-the-art in the PSP area and to point out the most remarkable advances already made, as well as to assist the methods' progress, directing to critical points that further can be tackled more productively (MOULT et al., 2018; KRYSHTAFOVYCH et al., 2019; KANDATHIL; GREENER; JONES, 2019). According to results related to the latest CASP editions (ABRIATA et al., 2018; XU; WANG, 2019; KRYSHTAFOVYCH et al., 2019), regarding the automatic FM category without manual intervention (servers), the methods related to the Baker group (OVCHINNIKOV et al., 2018) can be pointed out as some of the most popular performing methods, highlighting the method of Rosetta (ROHL et al., 2004).

Nevertheless, a great breakthrough[8][9] concerning the entire protein folding research field was recently announced, when the AI method developed by DeepMind[10] from Google AI has made an enormous advance in terms of the obtained results for the 3-D protein structure prediction problem only from the protein's amino acid sequence[11] (CALLAWAY, 2020). DeepMind's method, known as AlphaFold (JUMPER et al., 2020), significantly outperformed around 100 other research teams in the last CASP14 edition. The results[12] were announced on 30 November 2020, during the conference. According to AlphaFold's authors, this breakthrough demonstrates the impact AI can have on scientific discovery and its potential to accelerate progress in some of the most fundamental fields that explain real-life challenges.

[7]<www.predictioncenter.org>
[8]<https://www.nature.com/articles/d41586-020-03348-4>
[9]<https://www.sciencemag.org/news/2020/11/game-has-changed-ai-triumphs-solving-protein-structures>
[10]<https://deepmind.com/>
[11]<https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology>
[12]<https://predictioncenter.org/casp14/zscores_final.cgi>

4.7.1 Rosetta Method

Robetta[13] (KIM; CHIVIAN; BAKER, 2004) represents a major web server that uses the Rosetta prediction protocols[14] (ROHL et al., 2004; BRADLEY; MISURA; BAKER, 2005), both for *ab initio* predictions and comparative modeling (SONG et al., 2013). Rosetta is a fragment-based method (fragment assembly) (SIMONS et al., 1997), which uses small structural segments (3 and 9 amino acids) of known protein structures extracted from the PDB. Initial configurations of the target protein are created from the combination of a bunch of fragments. This method is divided into multiple optimization stages, where different structural representations and evaluation functions are employed (ROHL et al., 2004; LEAVER-FAY et al., 2013). The method starts with a low-resolution (coarse-grained) optimization and gradually increases the precision level until finalizing the process with a more accurate refinement technique (all-atom resolution), where the best-found structures are considered. We note that the complexity of energy functions follows the precision level's growth of the computational representation during the process. Through sampling and clustering of thousands of individuals, the method seeks to locate distinct conformations distributed on the energy landscape. The structural groups are optimized by several Monte Carlo simulations, known as Replica Exchange Monte Carlo (REMC), by exchange processes of simulation parameters and structural fragments.

4.7.2 AlphaFold Method

AlphaFold was first proposed in CASP13 (KRYSHTAFOVYCH et al., 2019; HUTSON, 2019), which achieved the highest accuracy among participants. AlphaFold (SENIOR et al., 2019; SENIOR et al., 2020) implements deep learning neural networks to structural and genetic data to predict the distance between pairs of amino acids in a protein, which conveys more information about the structure than contact predictions. Using this information, the authors constructed a potential of mean force based on physical and geometric constraints that can accurately describe the shape of a protein. A gradient descent algorithm optimizes this resulting potential to generate structures without complex sampling procedures.

[13]<www.robetta.bakerlab.org>
[14]<www.rosettacommons.org>

For the CASP14 edition, the AlphaFold version 2[15] (JUMPER et al., 2020) was presented. The authors created an attention-based deep learning neural network system different than the CASP13 AlphaFold. The novel version produces much more accurate protein structures and estimates of model accuracy. In this method, the folded protein is modeled as a "spatial graph", where residues are the nodes, while the edges connect the residues in close proximity. Thus, the neural network tries to interpret the structure of this graph while reasoning over the implicit graph that it is building. It uses evolutionarily related sequences, multiple sequence alignment (MSA), and a representation of amino acid pairs to refine the graph. By iterating this process, the system develops strong predictions of the protein's underlying physical structure and can determine highly-accurate structures. It should be noted that the CASP14 proceedings have not yet been published.

Despite the recent advances related to the PSP problem, we observe that this work is focused on the development and assessment of metaheuristics applied to hard problems, such as the PSP's multimodal energy function. Thus, the following section describes some metaheuristics and search algorithms applied to the PSP.

4.7.3 PSP Metaheuristics

Metaheuristics are techniques practically problem-independent and can be applied to a wide range of problems with no or minor modifications in their parameters (BOUSSAÏD; LEPAGNOT; SIARRY, 2013). Nevertheless, particularly in Structural Bioinformatics problems, the simple application of canonical methods is not always enough to achieve good performance. This fact is due to the high dimensionality of variables (BELDA et al., 2007; HANDL; LOVELL; KNOWLES, 2008). Thus, incorporating previous knowledge about the problem and exploring its specific characteristics can be seen as alternatives to increase the methods' effectiveness by limiting and reducing the solution space size (DORN et al., 2014).

Numerous problems from the most diverse knowledge areas encompass complex objective functions (GLIBOVETS; GULAYEVA, 2013). Regarding the PSP problem, the energy functions used as fitness functions in the 3-D protein structure optimization fit into the complex category of multimodal objective functions, where the same energy value may represent distinct conformations for the same target protein (HANDL; LOVELL; KNOWLES, 2008). Knowing the difficulties that energy functions have to

[15]<https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology>

represent global optima as the best solutions, it is interesting to discover the highest possible number of optimal conformations to provide sufficient resources for future specialist analysis (BELDA et al., 2007). For example, in the method of Rosetta, the final results of a prediction process consist not only of a single structural model but a set of energetically favorable and often topologically distinct solutions resulting from several optimization clusters used throughout the simulation process (ROHL et al., 2004).

As a consequence of the complexity of the PSP problem, a variety of metaheuristics were being proposed to deal with it (DORN et al., 2014). Such algorithms alter the 3-D structure orientation of the macromolecule under study through mathematical operations to minimize an energy function and approximate computational models to ideal solutions (CHOU, 2004; BRADLEY; MISURA; BAKER, 2005). For example, Elofsson et al. (ELOFSSON; GRAND; EISENBERG, 1995) proposed a GA combined with a heuristic responsible for performing small movements in dihedral angles of the protein structure, aiming the improvement of local minima exploitation.

Cutello et al. (CUTELLO; NARZISI; NICOSIA, 2006) developed a multi-objective EA to overcome the energy function inefficiencies. The authors have shown that the most common energy terms used in molecular interactions, bonded and non-bonded terms, are conflicting. Thus, these two types of terms were treated as distinct objectives in the optimization. Fonseca et al. (FONSECA; PALUSZEWSKI; WINTER, 2010) presented a variation of the bee colony optimization algorithm (KARABOGA; BASTURK, 2007). The algorithm was applied to the PSP for the first time, considering proteins more prominent than 50 amino acids. In the work of Dorn et al. (DORN; BURIOL; LAMB, 2011), a GA with a population based on "castes" was developed. It also uses the path relinking procedure as a LS. Saleh et al. (SALEH; OLSON; SHEHU, 2013) proposed a MA consisting of two evolutionary search strategies, based on structural fragments of amino acids, to address the multiple local minima problem in the energy function. For this, the authors worked on the modeling of two distinct energy functions, a modified version of the Associative Memory Hamiltonian with Water (AMW) function (SHEHU; KAVRAKI; CLEMENTI, 2009) and the centroid-based energy function of Rosetta (ROHL et al., 2004).

Regarding the knowledge exploration about aa conformational preferences, Dorn et al. (DORN et al., 2013) proposed a knowledge-based computational method whose objective is to reduce the search space size. Such an algorithm considers the aa conformational preferences based on previous occurrences of the target protein amino acids in experimentally determined proteins (LIGABUE-BRAUN et al., 2018). Borguesan et al. (BORGUESAN et al., 2015) demonstrate the Angle Probability List (APL) use, where the authors have shown the APL contributions by optimizing a set of 3-D protein structures using two different metaheuristics, which consisted of a GA and a PSO algorithm. In addition, through the webserver NIAS[16] (BORGUESAN; INOSTROZA-PONTA; DORN, 2016), the authors make available to the scientific community the creation of APLs to be used in structural modeling methods or in any other problem that may require insights about the aa conformational preferences. Also, in a previous work developed by the same authors (INOSTROZA-PONTA; FARFÁN; DORN, 2015), the first attempt to develop a MA that incorporates information from a variation of APL was made.

Related to multimodal optimization applied to the PSP, Garza-Fabre et al. (GARZA-FABRE et al., 2016) proposed a MA that associates the method of Rosetta as a LS heuristic. The authors developed specialized genetic operators by incorporating intrinsic knowledge about the problem, increasing the exploration of distinct conformations. As an alternative to energy function inaccuracies and search space roughness, they have used the stochastic ranking-based selection procedure, a multimodal technique, to minimize the evaluation function while maintaining the structural population diversity. Also, the method incorporates a modified version of the Rosetta-based population initialization to reach a proper balance between search space exploration and exploitation. The method uses the same energy functions and model representations used in the Rosetta optimization process.

Rocha et al. (ROCHA et al., 2016) proposed a multi-objective GA that uses the phenotypic crowding similarity metric for individual selection, where two solutions are selected according to their structural differences. In this work, the authors focused on reaching more diversified and well-distributed Pareto frontiers at the end of the optimization process by incorporating the crowding distance technique of the Non-dominated Sorting Genetic Algorithm (NSGA-II) (DEB et al., 2002) as the insertion criterion of new individuals in the population. Besides, the authors explored three classical force field potentials, three hydrogen bond potentials and a hydrophobic compaction term were combined in two configurations with different objectives for the fitness function.

Despite recent remarkable advances in the PSP area by the development of novel optimization techniques, mainly due to the discovery of experimental knowledge and incorporation into the prediction methods, such as the AlphaFold method, this research area

[16]<sbcb.inf.ufrgs.br/nias/>

still offers challenges. (KRYSHTAFOVYCH et al., 2019; XU; WANG, 2019; SHRESTHA et al., 2019; KANDATHIL; GREENER; JONES, 2019). The quality limitation of the structural fragment libraries and aa conformational preferences, energy function inaccuracies, and the high dimensionality of the conformational space make the PSP problem extremely complicated regarding the FM category from the metaheuristic's perspective (KIM et al., 2009; GARZA-FABRE et al., 2016).

In this way, search techniques as the main focus of improvements need to incorporate more robust mechanisms, which can keep energetically acceptable structures whereas correspond to distinct conformations distributed along with the search space's most favorable areas. The generation and preservation of a mixed solution set facing a multimodal problem determine factors for achieving competitive results, corroborating the issues previously discussed regarding global and multimodal continuous optimization for general purpose (DAS et al., 2011).

Therefore, in this thesis, we intend to develop an adaptive MA combined with an ABC algorithm used as an exploratory method with a LS strategy to enhance the method's performance. The algorithm is applied to single global and multimodal continuous benchmark functions and then applied to the PSP problem. It incorporates evolutionary and swarm intelligence concepts for global exploration and a LS strategy focused on refining the multimodal conformational space. The MA also uses the experimentally determined protein structure knowledge from the PDB through the aa conformational preferences and protein CM information.

4.8 Final Remarks

This chapter presented a review of the most relevant works related to metaheuristics for global and multimodal continuous optimization, such as classical and state-of-the-art algorithms. The chapter discussed existing literature concerns about search strategies and optimization, such as population diversity and convergence, Expr/Expt trade-off, parameter control and hybrid algorithms, and multimodal search strategies.

The chapter described an overview of computational methods and metaheuristics applied to the PSP problem. It introduced the CASP competition, the popular method of Rosetta, and the AlphaFold from DeepMind, which is state-of-the-art for protein folding.

Furthermore, we highlight that the general objective of this work is related to the investigation of distinct metaheuristic's aspects aiming at providing a metaheuristic for optimization problems regarding the multimodal continuous domain. With this, the methods in this thesis were designed based on an incremental approach, encompassing case studies of single global continuous optimization with multimodal functions, multimodal functions with many global optima, and the 3-D PSP problem. Thus, the methods and search strategies discussed in this chapter underlined relevant concepts used to develop this work.

5 MATERIAL AND METHODS

5.1 Introduction

As a baseline for the development of this work, we employed concepts of metaheuristics already presented in our previous works, which include (CORRÊA; INOSTROZA-PONTA; DORN, 2017; CORRÊA; DORN, 2018; CORRÊA; DORN, 2019; CORRÊA et al., 2020; CORRÊA et al., 2016; CORRÊA et al., 2018; CORRÊA; DORN, 2020). It should be observed that these papers were proposed to deal with the PSP problem. However, one of the goals of this thesis is to connect such ideas and apply them combined with novel strategies.

Therefore, our first objective was to present the most relevant works related to metaheuristics for multimodal continuous optimization and applied to the PSP, such as classical and state-of-the-art algorithms. Moreover, we intended to discuss the most crucial literature concerns about search strategies and optimization, such as population diversity and convergence, Expr/Expt trade-off, parameter control, hybrid algorithms, and multimodal strategies.

Following the first part of this thesis and the general objective delineated, our focus is to create a method by an incremental design, which can deal with the inherent multimodality and issues of a range of optimization functions while preserving accurate results. It is known that each optimization function requires distinct abilities from algorithms, even in the same application domain. Then, we expect to enhance our approach with such a need without degrading previously incorporated search strategies. For instance, adopting niching concepts, aiming to improve the search space exploration and reach a satisfactory trade-off between Expr/Expt of the individuals, while keeping the population diversity and enforcing the method's convergence when indicated by the search process' feedback. All of the proposed search strategies are detailed in the following sections.

5.2 Proposed Method for Single Global Optimization

The method proposed for continuous optimization is an adaptive framework-based MA, which incorporates concepts of bio-inspired algorithms for global optimization with separate local improvement. The MA is a hybrid metaheuristic composed of a population-based evolutionary framework and a LS algorithm used within the generation cycle of the external framework. As mentioned earlier (Section 4.4), MAs enable the combination of ideas from different search methodologies, which may provide better results than a single strategy to a given problem.

The method was implemented following the points of interest outlined in this thesis. It aims to efficiently explore the search space in an attempt to discover and maintain the search space solutions, keep a reasonable balance between Expr/Expt by controlling the population diversity, and control the parameter setting to better deal with global and multimodal optimization functions. Hence, we designed some MA versions to deal with global continuous optimization, then modified the resulting version for multimodal optimization, and finally tackled the PSP problem. Moreover, we observe that the intermediate versions developed to reach the final versions of the methods are also discussed in the section of computational experiments (Section 6.2).

5.2.1 Algorithmic Structure of the Method

The proposed metaheuristic was structured based on a hierarchical tree data structure implemented under a MA framework. This method consists of a multi-population technique, which arranges the population of individuals in subpopulations into the tree's nodes. The data structure adopted in this thesis was first proposed by Inostroza-Ponta et al. (INOSTROZA-PONTA; FARFÁN; DORN, 2015) and also explored in our previously proposed works (CORRÊA; INOSTROZA-PONTA; DORN, 2017; CORRÊA et al., 2020; CORRÊA et al., 2016; CORRÊA et al., 2018; CORRÊA; DORN, 2020). In these papers, the hierarchical tree was parameterized as a ternary tree. However, we emphasize that the papers mentioned above were focused only on the PSP problem. Moreover, we aim to connect the previously studied concepts with novel search ideas to create an optimization method that improves strategies aiming for better results.

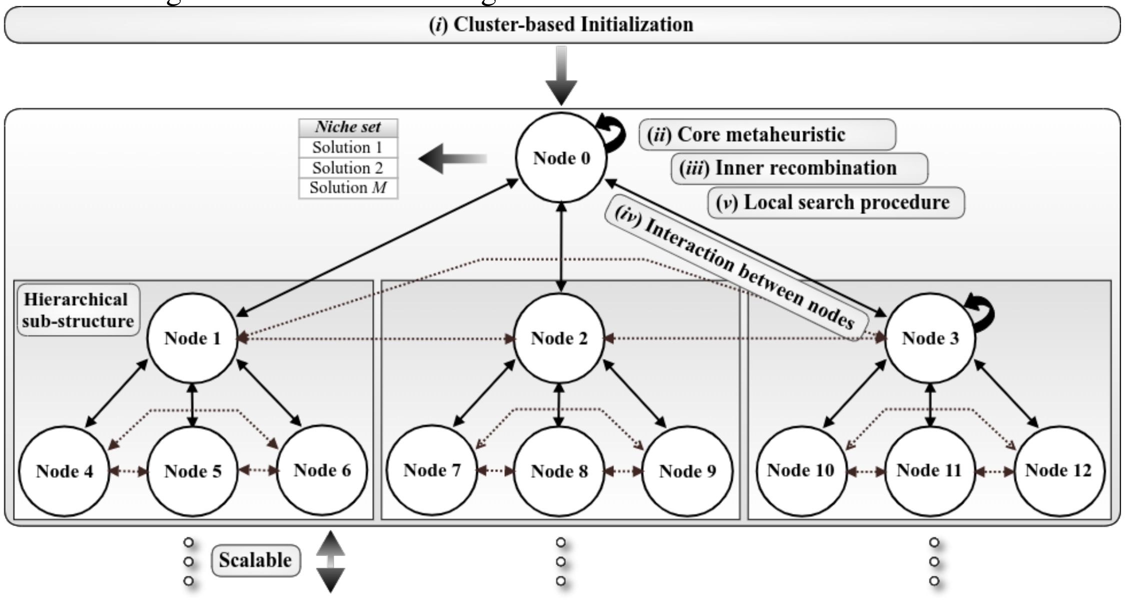
The MA population is organized into a ternary tree structure with N_{nodes} nodes, which characterize N_{nodes} subpopulations. The number of nodes is directly related to the number of layers (N_{layers}) in the tree, such that N_{nodes} is given by:

$$N_{nodes} = \sum_{i=0}^{N_{layers}-1} 3^i \tag{5.1}$$

Where 3^i represents the number of nodes in each layer of the ternary tree. In this approach, the tree structure is a perfect ternary tree, in which all internal nodes have three children, and all the leaf nodes are at the same depth or same layer.

Thus, each node receives part of the entire population, optimized internally by an independent core search metaheuristic. Throughout the optimization process, the nodes interact with each other, following predefined hierarchical rules, via crossover operations as a way of knowledge sharing, population diversification, and exploration inter-niches. This tree structure, which encompasses independent optimizations and interactions between distinct populations, was conceived as a niching strategy to overcome the multimodality issues over the evolutionary process and maintenance of the population diversity. With this, the algorithm can easily explore the search space, where better solutions located in different regions of the space can emerge. Figure 5.1 illustrates the proposed approach flowchart adopting three layers ($N_{layers} = 3$), which can be increased or reduced. Each MA component is further detailed in the following sections. Also, the different versions, parameterization, and particularities of the MA created by the constructive design are discussed in the section of computational experiments (Section 6.2).

Figure 5.1: Hierarchical organization of the framework-based MA



Source: From the author (2022).

5.2.2 Initialization of the Tree

The MA receives as input parameter the maximum number of individuals (Max_NS) allowed in the entire population of the tree structure. Thus, the Max_NS individuals are randomly initialized within the search space of each optimization function. To maintain a diverse population and overcome the genetic drift inherent to the population evolution, preventing a premature convergence to local optima, we adopted a cluster-based niching strategy to divide the entire population into subpopulations based on spatial positions of the individuals.

As already discussed in Section 2.3, multimodal strategies may increase the probability of a metaheuristic finding global optima since search efforts are not concentrated just in one region of the search space but in different areas. In our method, the niching strategy is used to increase population diversity by better populating the search space when dealing with single global optimization on a multimodal fitness landscape. It also aims to find all possible optimal or acceptable suboptimal solutions when optimizing multimodal functions with more than one global optimum.

The clustering procedure is incorporated in the Speciation niching algorithm (Section 4.6.1). As the species-based strategy involves a parameter of species distance (species radius), the clustering algorithm integrated into the classical niching strategies is used to mitigate the effects from the sensitivity to parameters. Thus, the clustering niching transfers the sensitive parameter (niche radius) to a less sensitive one (cluster size). As previously mentioned in Section 4.6.2 of related works, we observe that similar strategy was adopted in Qu et al. (QU; SUGANTHAN; LIANG, 2012), Gao et al. (GAO; YEN; LIU, 2013), Wang et al. (WANG et al., 2017), Wang et al. (WANG et al., 2019) and Liu et al. (LIU et al., 2021) as cluster-based SDE (Section 4.6.2).

The cluster-based niching strategy is a distance-based neighborhood algorithm that divides the NS individuals of population P into NCl subpopulations or niches, where each of which comprises M individuals with adjacent locations in the search space. A given solution is selected from the population as a seed to form a niche by including $M - 1$ individuals, which are closest to the seed given by Euclidean distance. These M individuals are then eliminated from the population. All the niches are formed according to this process. This is shown in Algorithm 2, which receives as input parameter the population P, the number of solutions NS to be clustered, and the number of clusters NCl. Regarding the MA framework, the number of clusters represents the number of nodes in the tree ($NCl = N_{nodes}$), and the population size is the maximum number of individuals allowed in the tree ($NS = Max\_NS$).

Algorithm 2 Pseudocode of the clustering-based Speciation algorithm.

```
Require: $P$: population of individuals; $NS$: population size; $NCl$: number of clusters
Ensure: $Clset$: set of the clusters generated
1: $SortPopulation(P)$ from best to worst according to the fitness values
2: $M \leftarrow NS/NCl$
3: for $i \leftarrow 1 : NCl$ do
4:     the best individual $X$ is set as the species seed
5:     $Clset_i \leftarrow X$ and its nearest $M-1$ individuals form a new cluster
6:     remove the $M$ individuals from $P$
7: end for
8: return $Clset$
```

After the niching procedure, the generated subpopulations are then incorporated into the tree's nodes. Thus, the number of niches is directly related to the number of nodes in the tree structure, where each subpopulation corresponds to a node of the MA with M solutions. Each niche is then optimized based on the MA search mechanisms. The optimization steps of the MA are described in the next section.

5.2.3 Optimization Steps of the MA

After the MA initialization, the steps below are executed in every algorithm generation. The stop criterion is determined by the maximum number of fitness evaluations (Max_Evls) performed during the optimization:

1. Each node runs independently the core search metaheuristic to optimize its solution set (subpopulation) (Figure 5.1-*ii*). The core search algorithm is executed by g_{core} generations in each node (Section 5.2.12);
2. Each node performs g_{core} inner recombination operations with the solutions in its respective subpopulation (Figure 5.1-*iii*). The offspring are inserted in the next population by the DC replacement strategy (Section 5.2.4). We note that the inner recombination is performed interspersed with the core search algorithm. At the end of each generation of the core search, the recombination is applied once;
3. The procedure of interactions between nodes is performed following the hierarchical structure of the tree, where each node only interacts with its neighbors from the same parent and with its parent (Figure 5.1-*iv*). Recombination operations do the

interactions between nodes, and the RTS strategy is used to insert the offspring in the population (Section 5.2.5);

4. The LS algorithm is applied via the procedure of local improvement on a given solution of each subpopulation to exploit further the search regions inside the niches (Figure 5.1-*v*) (Section 5.2.6);
5. The procedure of resizing the tree structure is applied to reduce the number of nodes linearly as a function of the number of fitness evaluations. The tree resizing is used to enforce the framework's ability for exploration at the early stages of the search process and enhance the exploitation at the final stages (Section 5.2.7);
6. The control mechanism concerning the convergence and performance of the algorithm is applied. It regulates them via indicators of population diversity, Expr/Expt efforts, and improvements of the solutions throughout the generations (Section 5.2.8);
7. The procedure to dynamically adjust the niche size and reorganize the solutions into the nodes according to their similarities through the clustering-based Speciation algorithm is performed (Section 5.2.9).

It should be noted that each step described above was sequentially implemented. However, the framework was designed to ease the implementation in distributed environments. Each node could run in a single core of a cluster and interact with other nodes through explicit messages, i.e., classical commands of send and receive, or shared memory approach. A similar tree structure was already presented in distributed systems as shown in our previous works (CORRÊA; INOSTROZA-PONTA; DORN, 2017; CORRÊA et al., 2020; INOSTROZA-PONTA et al., 2020). The MA procedures mentioned above are further detailed in the following sections.

5.2.4 Inner Node Recombination and Selection

The inner recombination strategy is performed at the end of every generation of the core metaheuristic inside each node. Regardless of the adopted core search algorithm, this operation aims to enhance the exploration around the delimited search space of a given subpopulation. As the proposed MA intends to provide a general optimization method that enables any core metaheuristic, the inner recombination is used to complement the search mechanisms of the framework, favoring the exploration of intra-niche regions. The recombination operation is performed by the Uniform crossover (SYSWERDA, 1989), one of the most used operators in GAs (MOSCATO; COTTA, 2019). The selection of individuals for crossover is made by a 3-way tournament selection strategy, where three individuals of the subpopulation are randomly selected, and tournament rounds are run among them. The fittest candidate amongst those selected individuals is chosen as a parent for the crossover. The 3-tournament selection was chosen due to its lower selection pressure characteristic when compared with other selection schemes, such as roulette wheel and ranking-based selection.

Thus, tournament selections select two solutions from the subpopulation as parents for the crossover. In the Uniform crossover, the offspring are created by randomly selecting one of the variables in any of the parents at each position. The selection of variables is uniform, where the variables are chosen from both parents with the same probability of 50%. After that, the offspring are inserted into the subpopulation via the DC replacement strategy (Section 4.6.1), where the two offspring generated are compared with their parents, and each offspring replaces the nearest parent by a greedy scheme (Section 4.6.1). DC replacement was adopted to maintain diversity and niche cohesion.

5.2.5 Interactions Between Nodes

The interactions between nodes are performed following the hierarchical structure of the tree, where each node only interacts with the ones in the same hierarchical sub-structure (Figure 5.1), i.e., with their neighbors from the same parent and with its parent. This hierarchical sub-structure can be divided into two levels: (*i*) upper level, which comprises a parent node with child nodes; and (*ii*) lower level, which includes the three child nodes of the parent node in the upper level. This division scheme is used to determine how the interactions between nodes occur in each sub-structure.

The interactions are done by recombination operations. These operations are also performed by the Uniform crossover, but the individuals for crossover are selected by the ranking-based selection, which consists of ranking solutions according to their fitness values, assigning to them ascending probabilities to be chosen based on the position of solutions. Thus, the best solution has the highest-ranking position, and the worst has the first position since individuals with better fitness values should be more likely to be selected. This strategy is used to favor the promising individuals of each subpopulation. The solutions are randomly selected based on the defined probabilities.

Then, the RTS strategy (Section 4.6.1) is used to insert the offspring in the subpopulation, where each offspring replaces its nearest solution by a greedy scheme. RTS was adopted to prevent different solutions from competing with each other in an attempt to maintain the cohesion intra-niche and the diversity inter-niches. The primary issue in RTS is the definition of the window size parameter (DAS et al., 2011). In our case, the window size parameter is the subpopulation size (M), as it means a portion of the entire population.

The MA performs two types of interaction inside a hierarchical sub-structure: (*i*) bottom-up crossover operation; and (*ii*) top-down crossover operation. These interactions are demonstrated in the Algorithm 3, which receives as input parameter the index of the parent node p_{index} of the hierarchical sub-structure submitted to the operations.

Algorithm 3 Pseudocode of the bottom-up and top-down interactions between nodes.

```
Require: $p_{index}$: index of the parent node of the hierarchical sub-structure
Require: $niche$: set of the niches
Ensure: $niche$: set of the niches after the recombination operations regarding the $p_{index}$
     //Selection and bottom-up crossover
 1:  $set_{offspring}[3] \leftarrow \emptyset$
     //Crossover between child nodes
 2:  $parent_1 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+1}$
 3:  $parent_2 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+2}$
 4:  $set_{offspring}[0] \leftarrow UniformCrossover(parent_1, parent_2)$
 5:  $parent_1 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+1}$
 6:  $parent_3 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+3}$
 7:  $set_{offspring}[1] \leftarrow UniformCrossover(parent_1, parent_3)$
 8:  $parent_2 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+2}$
 9:  $parent_3 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+3}$
10:  $set_{offspring}[2] \leftarrow UniformCrossover(parent_2, parent_3)$
11:  $niche_{p_{index}} \leftarrow RTS(niche_{p_{index}}, set_{offspring})$
     //Selection and top-down crossover
     //Crossover between parent node and children
12:  $parent_{node} \leftarrow$ ranking-based selection of solution from $niche_{p_{index}}$
13:  $parent_1 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+1}$
14:  $offspring \leftarrow UniformCrossover(parent_{node}, parent_1)$
15:  $niche_{p_{index}*3+1} \leftarrow RTS(niche_{p_{index}*3+1}, offspring)$
16:  $parent_{node} \leftarrow$ ranking-based selection of solution from $niche_{index}$
17:  $parent_2 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+2}$
18:  $offspring \leftarrow UniformCrossover(parent_{node}, parent_2)$
19:  $niche_{p_{index}*3+2} \leftarrow RTS(niche_{p_{index}*3+2}, offspring)$
20:  $parent_{node} \leftarrow$ ranking-based selection of solution from $niche_{index}$
21:  $parent_3 \leftarrow$ ranking-based selection of solution from $niche_{p_{index}*3+3}$
22:  $offspring \leftarrow UniformCrossover(parent_{node}, parent_3)$
23:  $niche_{p_{index}*3+3} \leftarrow RTS(niche_{p_{index}*3+3}, offspring)$
24:  return $niche$
```

Bottom-up crossover operation: The child nodes (lower level) of the same hierarchical sub-structure do crossover operations with their neighbors, linked to the same parent (upper level), to generate three offspring for each group of neighbors (Algorithm 3, lines 2-10). For instance, the hierarchical sub-structure of node 0 as the parent includes nodes 1, 2, and 3 as children, where the recombination process occurs between nodes 1-2, 1-3, and 2-3, as illustrated by the dashed arrows connecting the child nodes in Figure 5.1. One solution from each subpopulation is selected by ranking-based selection strategy as parents in each crossover operation. Thus, the resulting offspring from the crossovers between the child nodes are integrated into the population of the respective parent by the RTS strategy (Algorithm 3, line 11). For instance, in the hierarchical sub-structure of node 0, the three generated offspring are inserted into its subpopulation.

Top-down crossover operation: In a similar process as described above, the parent nodes (upper level) perform top-down crossover operations with their children (lower level) to generate three offspring. A total of three operations are performed inside a hierarchical sub-structure since each operation results in an offspring, where each one is generated by the crossover between the parent and each one of its children (Algorithm 3, lines 12-23). The resulting offspring from the crossover operation between each parent and its children are integrated into the respective child subpopulation by the RTS strategy. For instance, in the hierarchical sub-structure of node 0 as the parent, the recombination process occurs between nodes 0-1, 0-2, and 0-3, as illustrated by the black arrows connecting the parent and child nodes in Figure 5.1. Each offspring is inserted in the respective subpopulation of the child nodes.

This interaction mechanism among nodes can increase the MA's exploratory potential, as solutions in different regions can search for better points located between them but out of the niches. The two types of recombination were designed to integrate all the nodes as a way of knowledge sharing inter-niches and population diversification, so that the information of each one can eventually be passed to all the nodes throughout the evolutionary process. This mechanism is especially important in single global optimization, where the niches should approach the same optimum during the optimization process.

5.2.6 Local Improvement Procedure

The LS algorithm is applied periodically on the best solutions of each subpopulation regarding the local improvement procedure (LIP). We created the LIP to regulate the

MA's exploration through the search space and the refinement of the promising individuals according to the fitness function evaluations utilized by each of these mechanisms. The LIP uses the continuous LS strategy, which consists of a chained LS to adjust the exploitation intensity (number of fitness evaluations) applied to the MA throughout the evolutionary process. Such strategy aims to refine the most promising solutions maintaining the historical memory of the LS procedures already performed on each individual. This strategy of LS chain was presented in the work of Molina et al. (MOLINA; LOZANO; HERRERA, 2010) and also used in our previous work (CORRÊA et al., 2018).

In this work, we designed the MA framework combined with the SW algorithm (SOLIS; WETS, 1981) as its LS strategy. The LS chain is used to control the SW parameters for each solution and the execution period of the LS. It should be observed that the SW algorithm can be replaced by any LS algorithm, preserving the LIP's mechanism.

Solis and Wets algorithm: The SW metaheuristic is a randomized hill-climber heuristic with an adaptive step size which starts at a given point x (objective variable of an individual) of the search space. A constant of deviation d is randomly created under a normal distribution with standard deviation p. If $(x + d)$ or $(x - d)$ improve the current step x, a move is performed to the better x, and *success* is recorded. Otherwise, *failure* is recorded. The adaptive step of the algorithm is defined by adjusting the parameter p according to the number of successes and failures obtained over the search. After a given number of successes ($maxSuccesses$), p is increased to raise the step size of the search, and after a given number of failures ($maxFailures$), p is decreased to constraint the search. Also, a bias term b is used to guide the method towards the successes.

LS chain strategy: The chained LS enables that the same individual becomes the starting point of subsequent applications of the LS, as the best solutions may exist for several generations in the population. The LS chain strategy keeps the history of the LS parameterization of each individual to be used as the initial parameter values in the next LS application, providing an uninterrupted connection between successive LS invocations of the same individual. Thus, it uses a self-adapting scheme to control the parameters of the SW algorithm. Each $individual_i$ in the population of the MA has its own parameter values of $maxSuccesses_i$, $maxFailures_i$, p_i and b_i, initialized equally. After the refinement of an individual, the parameters that define the current state of the LS process are stored along with it. When this individual is selected again to the LS, the previous control parameters of the SW algorithm are used.

LS application: The LS procedure uses a constant to regulate the LS intensity

(I_{str}) every time that the SW algorithm is applied. The LS intensity is defined according to the total number of fitness function evaluations performed in one execution of the SW. Based on this, Molina et al. (MOLINA; LOZANO; HERRERA, 2010) created the ratio parameter ($r_{L/G}$) responsible for balancing the exploration efforts on global search and the refinement of the most promising regions by LS, preventing unnecessary exploitation. Hence, for every $n_{frec} \times N_{nodes}$ (Equation 5.2) number of fitness evaluations of the MA (global search), the LIP is applied to one of the $p_{best}\%$ best solutions of each subpopulation in an attempt to enhance the exploitation around the niches or escape from local minima (local search). With this, the SW algorithm is executed on each individual for I_{str} number of fitness evaluations after $n_{frec} \times N_{nodes}$ fitness evaluations are executed in the MA.

$$n_{frec} = I_{str} \frac{1 - r_{L/G}}{r_{L/G}} \tag{5.2}$$

For each $node_i (i = 1, \cdots, N_{nodes})$ in the tree structure, the $p_{best}\%$ best solutions of the $node_i$ are stored into its set of best solutions S_{best_i}. Starting the selection from the best to the worst individual in S_{best_i}, it is submitted to the SW algorithm if: (*i*) it has never been optimized; or (*ii*) it was previously refined with success, i.e., its fitness value was improved by the SW in the last application. The LS is applied to one of the best individuals that satisfy these conditions. For instance, if the best solution is not selected for LS, the second best is tested for submission, and so on. We note that only one individual of each niche is refined in each period that the LIP is executed.

LS restarting procedure: Suppose any of the $p_{best}\%$ best solutions of $node_i$ in S_{best_i} is submitted to the SW algorithm. In that case, the LS restarting procedure restarts its subpopulation, only keeping the best solution, and generates a new one. The LS restarting is used to diversify the population since if none of the $p_{best}\%$ best solutions of the niche has not been improved by the LS, this indicates that the subpopulation is possibly in a degree of convergence to a local optimum. Whenever a node is restarted, the clustering-based Speciation algorithm is executed through the dynamic niche size procedure (DNSP) to reorganize the solutions into the nodes according to their similarities and adjust the niche size. The niche size is changed whether no solution of any node is submitted to the LS (Section 5.2.9).

Parameterization of the local improvement procedure: We used $maxSuccesses = 5$, $maxFailures = 3$, $p = 1.0$ and $b = 0$ for initial control parameter values of the SW as indicated in Molina et al. (MOLINA; LOZANO; HERRERA, 2010) and Corrêa et al. (CORRÊA et al., 2018). The control parameters for Expr/Expt were defined as $I_{str} = 1000$ fitness function evaluations and $r_{L/G} = 0.5$, consequently $n_{frec} = 1000$ fitness evaluations (Equation 5.2). This means that the MA and the LS are alternatively applied every $N_{nodes} \times 1000$ fitness function evaluations and that each phase is assigned 50% of the total number of evaluations. For the $p_{best}\%$ control parameter, we adopted $p_{best}\% = 25$, which refers to the 25% best solutions of each subpopulation.

5.2.7 Tree Resizing Procedure

As a strategy to enforce the algorithm's exploration at the beginning of the search process and reach a reasonable convergence at the last generations of the optimization, we designed a linear reduction of the number of nodes in the tree structure in the function of the number of fitness evaluations. This scheme was inspired by the LPSR of the L-SHADE, previously explained in Section 4.5.3. The tree resizing procedure (TRP) is a deterministic algorithm that shrinks the number of layers in the tree linearly according to the number of fitness function evaluations.

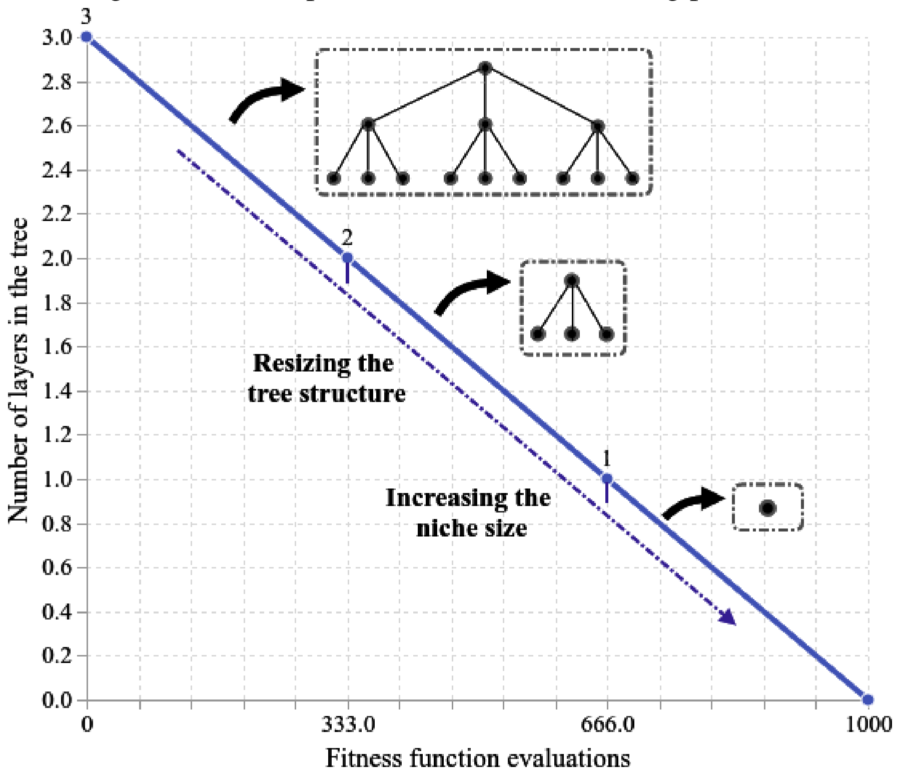
TRP reduces the tree structure removing layer by layer to preserve the perfect ternary tree throughout the process. Thus, given the maximum number of fitness evaluations (Max_Evls) and the initial number of layers ($N\_initial_{layers}$), the TRP removes one layer of the tree at every rm_{layer} number of fitness function evaluations of the framework. rm_{layer} is defined as follows:

$$rm_{layer} = \left(\frac{Max\_Evls}{N\_initial_{layers}} \right) \tag{5.3}$$

In this scheme, the outermost layer of the tree (the leaf nodes) is permanently removed. At the end of the optimization process, the MA will have precisely one layer with a single node, i.e., the root node 0 in Figure 5.1. Thus, the dynamic tree structure scheme combined with the clustering-based niching strategy aims to promote population diversity at the early optimization stages. However, throughout the evolutionary process, it modifies the tree structure, reducing the number of nodes, to increasingly focus on the most promising regions found. Figure 5.2 illustrates the TRP adopting $Max\_Evls = 1000$ and $N\_initial_{layers} = 3$.

After the resizing of the tree, the clustering-based Speciation algorithm is applied through the DNSP to reorganize the subpopulations and adjust the niche size (Sec-

Figure 5.2: Exemplification of the tree resizing procedure



Source: From the author (2022).

tion 5.2.9). As already mentioned, the MA receives as input parameter the maximum number of individuals (Max_NS) in the entire population, where the Max_NS remains unchanged during the method's execution. According to Algorithm 2, the number of clusters NCl is the current number of nodes in the tree after the TRP, and the population size remains the initial maximum number of individuals allowed in the tree ($NS = Max\_NS$). With this, the number of solutions in each subpopulation increases each time the TRP is performed. This strategy aims to favor the exploration around these regions by increasing the subpopulation size.

However, the reduction of the niches by the TRP and the increase of the subpopulation size can lead the optimization to some counter issues, such as loss of population diversity or slow convergence rate, depending on the state of the search process (i.e., the exploration effort measured by the population diversity degree and the performance of the algorithm at improving the best solutions). To regulate such issues, we implemented a control mechanism concerned with the convergence and performance of the algorithm. This strategy is described in the next section (Section 5.2.8).

5.2.8 Control Procedure for Convergence and Performance

According to Sections 2.4 and 2.5, the diversification of population is required to ensure a suitable algorithm's exploration of the search space and the ability to overcome local optima (BOUSSAÏD; LEPAGNOT; SIARRY, 2013; SER et al., 2019). Contrarily, the loss of population diversity in a given optimization period is necessary for the exploitation and convergence of the algorithm (ČREPINŠEK; LIU; MERNIK, 2013). Consequently, the metaheuristic can reach better performance in terms of optimization results when an appropriate balance between Expr/Expt is achieved. This means that a good balance may indicate reasonable exploration, exploitation, and convergence of the metaheuristic. On the other hand, a non-satisfactory ratio, at the wrong time, may increase the probability of premature convergence or deteriorate the algorithm's convergence speed (GUPTA; GHAFIR, 2012; MORALES-CASTAÑEDA et al., 2020).

However, it is known that the Expr/Expt ratio is problem-dependent (XU; ZHANG, 2014). Thus, considering that metaheuristics present a wide range of distinct search components and strategies, it is hard to find a general balance for all methods since their components and parameters are responsible for determining this ratio throughout the process (MORALES-CASTAÑEDA et al., 2020).

Regarding the search mechanisms adopted in the proposed framework (Section 5.2.3), we employed the MA with the tree structure and its procedures to encompass some adaptive and hybrid approaches to force the trade-off with specialized search operators in both discovery and refinement of solutions. For instance, to better populate the search space, increase population diversity at the early stages of the process, and favor exploration, we adopted the clustering-based niching initialization. The core search metaheuristic and inner recombination are used to explore the intra-niche regions. The inner recombination employs tournament selection and DC replacement strategies to maintain the diversity intra-niche regardless of the chosen core algorithm. Nevertheless, the interactions between nodes were designed to explore the inter-niche areas and improve the quality of subpopulations by sharing promising information from the entire tree. The RTS was employed to keep the feature and cohesion of each niche. Besides, a local improvement procedure was implemented to force the trade-off between exploration, through the strategies aforementioned, and exploitation of the best individuals by the SW algorithm as the LIP alternates between rounds of global and local searches. Hence, as the population-based algorithms tend to converge towards an optimal point, we designed the TRP to change the tree structure dynamically during the optimization process through a deterministic scheme. TRP reduces the number of nodes to focus on promising regions while increasing the number of solutions in each subpopulation to avoid losing diversification entirely and no longer searching around them.

Despite the strategies adopted in our method to enforce both Expr/Expt, we designed a procedure to quantify the balance between these two complementary goals throughout the optimization by controlling the population diversity degree with diversity measures and the algorithm's performance at improving the best solutions. The control mechanism aims to avoid possible issues from the synergy among the search components over distinct objective landscapes, such as slow convergence rate concerning the MA data structure and the niching strategies adopted or premature convergence caused by the chosen core metaheuristic, the LS algorithm, and the interactions between nodes.

As mentioned in Section 2.5, diversity measures are problem-dependent, and most of the existing search algorithms that focus on explicit diversity control tend to use them concerning the objective or decision space. Thus, we employed a strategy that combines the diversities measured in both the objective and decision spaces to quantify the search space and the distribution of individuals. The diversity measure used in this work represents the average of four distinct diversity measures, previously presented in Section 2.5, of which two distance-based measures: (*i*) the dimension-wise diversity measure ($Div_{dimension}$) given by Equation 2.1; and (*ii*) the population center measure (Div^2_{center}) described in Equation 2.3. And two other measures focused on the objective space: (*iii*) the entropy-based measure ($Div_{entropy}$) given by Equation 2.4; and (*iv*) the fitness-based distance ($Div_{fitness}$) showed in Equation 2.5. Such a combination of measures was adopted in an attempt to favor their strengths and overcome their shortcomings to distinguish distinct populations over the search space and provide better guidance for the process. With this, the diversity measure incorporated in the MA is given by Equation 5.4 and denotes the average of the normalized values of the four diversity measures considered.

$$Div_{combined} = \left(\frac{norm(Div_{dimension} + Div^2_{center} + Div_{entropy} + Div_{fitness})}{4} \right) \tag{5.4}$$

Where $norm$ is the normalization scaling method used to rescale the measures to a standard range through the initial diversity value of each measure calculated from the first initialized population.

Therefore, to quantify the Expr/Expt ratio through the population diversity, we adopted one of the definitions already presented in Section 2.6, which represents the balance as the percentage of Expr/Expt performed by a given metaheuristic (MORALES-CASTAÑEDA et al., 2020). Considering the combined diversity measure in Equation 5.4, the Expr/Expt ratio is defined by the relative diversity of the population, as follows:

$$Expr\% = \left(\frac{Div_{combined}}{Div\_initial_{combined}} \right) \tag{5.5}$$

$$Expt\% = \left(\frac{\mid Div_{combined} - Div\_initial_{combined} \mid}{Div\_initial_{combined}} \right) \tag{5.6}$$

Where $Div_{combined}$ is the diversity rate of the current iteration and $Div\_initial_{combined}$ is the initial diversity value of $Div_{combined}$ calculated from the first initialized population. We note that $Div\_initial_{combined}$ tends to be one of the highest diversity values found throughout the process. The percentage of exploration ($Expr\%$) describes the exploration effort as the relationship between the diversity in each iteration and the initial diversity degree. The percentage of exploitation ($Expt\%$) is calculated as the complementary percentage to $Expr\%$ and describes the exploitation effort. It should be noted that both $Expr\%$ and $Expt\%$ are mutually complementary, then addressing one of them is enough to deal with both of the efforts.

In this work, the $Expr\%$ measure is employed to evaluate the relationship between the population diversity, the Expr/Expt balance, and the metaheuristic's performance at improving the best solutions. The control mechanism for convergence and performance of the algorithm can be divided into two different procedures: (*i*) the restarting procedure; and (*ii*) the performance verification.

Restarting procedure: The restarting is considered one of the essential components of a MA, mainly used to avoid premature population convergence and support the method to escape from local minima (MOSCATO; COTTA, 2010). In the proposed MA, the niche restarting procedure is responsible for monitoring and independently restarting each subpopulation of the tree if it reaches a premature convergence according to the $Expr\%$ values (Equation 5.5).

With this, the population of a given node is reinitialized if it has reached $Expr\%$ less than $p_{expr}\%$ and if the simple moving average (SMA) of the $Expr\%$ values shows a decreasing tendency when compared with the SMA calculated in the previous period of generations. $Expr\%$ values are expressed as a percentage in the continuous range $[0, 1]$, and the parameter $p_{expr}\%$ is the threshold used to indicate whether the subpopulation has signs of convergence or not, as lower values of $Expr\%$ indicate low algorithm's exploration.

The SMA represents an arithmetic moving average calculated by adding current values of interest and then dividing that sum period by the number of generations in the calculation average. In our case, the SMA is used to analyze the $Expr\%$ values by creating two comparison periods of averages. These periods encompass a certain number of overlapped generations, where the latest period may drop the earliest $Expr\%$ value (oldest generation) and includes the latest $Expr\%$ (current generation), similar to a FIFO (First in, First out) schema. For instance, a x-generation moving average would average the closing $Expr\%$ values for the first x generations as the first data point. The next data point would drop the earliest $Expr\%$, add the $Expr\%$ on generation $x + 1$, and then take the average, and so on. The SMA of the current and the past periods are compared to follow the exploration behavior over these generations.

The latest period is the current SMA_Expr_{curr}, which considers the current value of $Expr\%$ in the current generation and the last $n\_gen - 1$ values measured in the last $n\_gen - 1$ generations. The second period is the $SMA\_Expr_{previous}$, which means the last SMA calculated before the current SMA. The $SMA\_Expr_{previous}$ considers the last n_gen values of $Expr\%$ calculated before the last (current) generation included in the current SMA_Expr_{curr}. The SMA_Expr_{curr} and $SMA\_Expr_{previous}$ are given by Equations 5.7 and 5.8, respectively.

$$SMA\_Expr_{curr} = \frac{1}{n\_gen} \sum_{i=gen_{curr}-(n\_gen-1)}^{gen_{curr}} Expr\%_i \tag{5.7}$$

$$SMA\_Expr_{previous} = \frac{1}{n\_gen} \sum_{i=(gen_{curr}-1)-(n\_gen-1)}^{gen_{curr}-1} Expr\%_i \tag{5.8}$$

Where n_gen is the period in terms of generations used to calculate the SMA and gen_{curr} represents the current generation of the algorithm's execution. The SMA was used with the $Expr\%$ values to identify the exploration variation through the optimization process. The moving average can smooth out short-term fluctuations of the measure and highlight trends of variations instead of using only the $Expr\%$ of the current generation compared to the previous one. For instance, the framework can be used with any core metaheuristic with distinct diversification strategies and selection pressure. The SMA can better describe the increasing/decreasing exploration trend instead of the single comparison between the current status of exploration and the previous one.

Thus, for each $node_i (i = 1, \cdots, N_{nodes})$ in the tree structure, its subpopulation is reinitialized if:

1. $Expr_{curr_i} < p_{expr}\%$, which indicates that the current $Expr\%$ is less than $p_{expr}\%$;
2. $SMA\_Expr_{curr_i} < SMA\_Expr_{previous_i}$, which indicates that the current SMA of $Expr_i\%$ values is less than the SMA of n_gen generations ago (i.e., the exploration is decreasing according to a period of generations).

If restarting, the procedure discards the entire subpopulation and generates a new one, only keeping the best solution. The restarting procedure is shown in Algorithm 4, which receives as input parameters the subpopulations of nodes $nodes$, the initial diversity of each one $Div\_initial_{combined}$ and the period in terms of generations n_gen used to calculate the SMA.

Algorithm 4 Pseudocode of the restarting procedure.

```
Require: $nodes$: node subpopulations; $Div\_initial_{combined}$: initial diversities of $nodes$; $n\_gen$:
    number of generations used to calculate the SMA
Ensure: $nodes$: node subpopulations after the restarting procedure
 1: $p_{expr}\% = 0.01$
 2: for each $node_i$ in $nodes$, $i \leftarrow 1 : N_{nodes}$ do
      //Population diversity and exploration inference
 3:     calculate the current $Div_{combined_i}$ for $node_i$ (Equation 5.4)
 4:     calculate $Expr_{curr_i}$ from $Div\_initial_{combined_i}$ and $Div_{combined_i}$ for $node_i$ (Equation 5.5)
 5:     store $Expr_{curr_i}$ to be considered in the SMA of $Expr\%$ values
 6:     $SMA\_Expr_{previous_i} \leftarrow SMA\_Expr_{curr_i}$
 7:     $SMA\_Expr_{curr_i} \leftarrow$ calculate SMA of $Expr\%$ values through $n\_gen$ generations for $node_i$
    (Equation 5.7)
 8:     if $(Expr_{curr_i} < p_{expr}\%)$ and $(SMA\_Expr_{curr_i} < SMA\_Expr_{previous_i})$ then
          //Exploration is decreasing
 9:         restart the subpopulation of $node_i$
10:     end if
11:     restart the $Expr\%$ values stored
12: end for
13: return $nodes$
```

The restarting is used to diversify the population since a low $Expr\%$ rate means a low population diversity. Depending on the value adopted for the threshold $p_{expr}\%$, the $Expr\%$ may represent the convergence of the population to a local optimum. Thus, recommended values for $p_{expr}\%$ are in the continuous range $[0, 0.1]$ (MORALES-CASTAÑEDA et al., 2020). In this work, we adopted $p_{expr}\% = 0.01$, which means that the restarting is performed in a given niche if the exploration rate is less than 1%. This low value was adopted to guarantee a probable population convergence since the framework has other mechanisms to slow down the convergence and not only the restarting. The comparison between the current SMA and the previous one was included to verify the loss of diversity in a given niche, which combined with the low exploration rate corroborates with the convergence of the subpopulation. Moreover, it should be noted that the increase of diversity given by the comparison of the two periods of SMAs, even with $Expr\%$ being less than $p_{expr}\%$, could indicate that diversity was recently inserted into the niche. Then, the MA can still find a promising path of improvement from this diversity, and restarting is avoided.

Finally, whenever a node is restarted, the clustering-based Speciation algorithm is executed through the DNSP to reorganize the solutions into the nodes according to their similarities (Section 5.2.9). However, we observe that the niche size is not changed when the niche restarting is performed.

Performance verification procedure: The procedure aims to control the performance of the MA by analyzing the number of improvements of solutions throughout a certain period of generations. Similar to the restarting, but less aggressively, it intends to avoid the method's stagnation and support the algorithm to escape from local minima by dynamically changing the size of each niche depending on the exploration status at a given moment of the algorithm's execution. The procedure works using a dynamic threshold ($evol_{time}$) related to the period with no improvement (not_imp) of the gbest solution.

Firstly, the variable not_imp is initialized with 0 and is incremented by 1 with each consecutive generation that the gbest solution is not improved, i.e., the fitness value of the gbest remains the same as the previous generation. Then, if the gbest solution has not been improved during $evol_{time}$ number of generations (i.e., $not\_imp > evol_{time}$), this procedure applies the DNSP in the entire population of the MA. DNSP is responsible for changing the size of subpopulations considering the increasing/decreasing of the exploration given by the SMA of two periods of $Expr\%$ values and reorganizing the individuals through the clustering-based niching algorithm. Such procedure is explained in the next section (Section 5.2.9).

Secondly, to update the dynamic threshold $evol_{time}$, the difference between the number of current generation (gen_{curr}) and the previous threshold used in the last update of the procedure ($evol_{previous}$) is stored as the new value for the threshold (i.e., $evol_{time} =$

$gen_{curr} - evol_{previous}$). With this, whenever the condition ($not\_imp > evol_{time}$) is satisfied, the threshold is updated as the number of generations that the MA took to reach the no improvement threshold since the last time that this condition was satisfied. The variable $evol_{previous}$ gets the previous $evol_{time}$ value. The variables $evol_{time}$ and $evol_{previous}$ are initialized with the low value of 1 in an attempt to quickly adjust the size of subpopulations to suitable search indicators at the beginning of the process, in case of no improvement of the best solution. Besides, it should be noted that the threshold $evol_{time}$ increases according to the number of improvements performed by the algorithm. The more improvements without satisfying the condition for the application of the DNSP, the higher the threshold $evol_{time}$ when updated.

The Algorithm 5 shows the pseudocode of the performance verification procedure. It receives as input parameter the MA population, current value of the not_imp variable, the current dynamic threshold $evol_{time}$, the previous dynamic threshold $evol_{previous}$, and the current generation gen_{curr}.

Algorithm 5 Pseudocode of the performance verification procedure.

```
Require: $P$: MA population; $not\_imp$: number of generations that gbest solution has not
    been improved so far; $evol_{time}$: current dynamic threshold; $evol_{previous}$: previous dy-
    namic threshold; $gen_{curr}$: current generation
Ensure: $P$: MA population after the performance verification procedure
1: if $not\_imp > evol_{time}$ then
2:     $P \leftarrow$ apply the dynamic niche size procedure in the MA population (Algorithm 6)
3:     $evol_{time} = gen_{curr} - evol_{previous}$
4:     $evol_{previous} = gen_{curr}$
5:     $not\_imp = 0$
6: end if
7: return $P$
```

5.2.9 Dynamic Niche Size Procedure

One of the most critical parameters to adapt is the population size since such parameter strongly influences the balance between Expr/Expt (SER et al., 2019). It is known that the optimal population size is problem-dependent. Nevertheless, combining strategies to adapt this parameter over the search process with other sensitive issues to guide better the method, such as the Expr/Expt trade-off and the algorithm's performance at improving the best solutions, can enhance the final results (ALETI; MOSER, 2016; POLÁKOVÁ; BUJOK, 2018; SER et al., 2019). Thus, the proposed MA already adopts the TRP to dynamically reduce the number of nodes in the tree structure to favor the exploration at the early stages of the optimization process and ease the exploitation around the promising regions at the last stages. Moreover, the Expr/Expt balance inferred in terms of the population diversity and the algorithm's performance is also addressed by the LS restarting, niche restarting, and performance verification procedures.

Therefore, as a complement to these search procedures, we designed the dynamic niche size procedure (DNSP) to dynamically adapt the size of the tree structure's subpopulations and reorganize the individuals through the clustering-based niching algorithm.

This procedure changes the size of subpopulations regarding the minimum and maximum (Min_NS_{node}, Max_NS_{node}) number of individuals allowed in each node. The Min_NS_{node} is defined according to the minimum number of solutions required in the core metaheuristic. In this work, we adopted $Min\_NS_{node} = 5$ which is suitable for most population-based metaheuristics. The Max_NS_{node} is defined based on the ratio between the input parameter that defines the maximum number of individuals (Max_NS) in the entire population of the framework and the number of nodes (N_{nodes}) in the tree structure given by Equation 5.9. The N_{nodes} is given by the number of layers (N_{layers}) in the tree (Equation 5.1). We observe that the number of nodes in the tree decreases whenever the TRP is applied, and hence the Max_NS_{node} is increased, as previously discussed.

$$Max\_NS_{node} = \frac{Max\_NS}{N_{nodes}} \tag{5.9}$$

The DNSP is applied whenever: (*i*) the TRP (Section 5.2.7); (*ii*) the LS restarting (Section 5.2.6); (*iii*) the niche restarting (Section 5.2.8); and (*iv*) the performance verification procedure (Section 5.2.8) are executed. However, the conditions to change the subpopulation size in DNSP are different in each case. Such conditions are described as follows. Also, these procedures are executed following priority orders, discussed below (Section 5.2.10).

First scenario: Whenever the TRP procedure is executed, the DNSP is performed. When this procedure is performed, the new size of subpopulations is defined randomly based on the minimum (Min_NS_{node}) and maximum (Max_NS_{node}) number of individuals allowed in each node. Thus, the new subpopulation size M after the TRP application is given by $M_{new} = rand(Min\_NS_{node}, Max\_NS_{node})$, where $rand$ generates an integer number in the range $[Min\_NS_{node}, Max\_NS_{node}]$.

The size of subpopulations is randomly defined in this scenario because the TRP changes the tree structure and modifies the maximum number of individuals (Max_NS_{node})

allowed in each node. Then, any condition based on the exploration degree would be affected.

Second scenario: Whenever the LS restarting and the performance verification procedures are performed, the DNSP dynamically changes the size of subpopulations of the tree structure, considering the variation in the increase and decrease of the exploration given by the SMA of two distinct periods of $Expr\%$ values. With this, the procedure adapts the subpopulation size regarding two significant issues of the optimization process, i.e., the exploration through the population diversity degree and the performance via the improvement of best solutions. However, we note that the niche size is only changed in the LS restarting whether no solution of any node is submitted to the LS.

Therefore, the DNSP considers the average $Expr\%$ of the entire MA population, i.e., all subpopulations of the tree. It calculates the average $Expr\%$ regarding all nodes of the tree in each generation ($Expr_{avg}\%$), as follows:

$$Expr_{avg}\% = \frac{1}{N_{nodes}} \sum_{i=1}^{N_{nodes}} Expr_i\% \tag{5.10}$$

Where $Expr_i\%$ represents the exploration given by Equation 5.5 of each $node_i(i = 1, \cdots, N_{nodes})$ in the tree structure.

Then, the SMA is used to analyze the $Expr_{avg}\%$ values by creating two comparison periods of averages. We highlight that the two periods of generations are calculated in the same way as for the restarting procedure, but using the $Expr_{avg}\%$ values of each generation instead of considering each node individually. The latest period is the current $SMA\_Avg\_Expr_{curr}$ (Equation 5.7), which considers the current value of $Expr_{avg}\%$ in the current generation and the last $n\_gen - 1$ values measured in the last $n\_gen - 1$ generations. And the second period is the $SMA\_Avg\_Expr_{previous}$ (Equation 5.8), which means the last SMA calculated before the current SMA. The $SMA\_Avg\_Expr_{previous}$ considers the last n_gen values of $Expr_{avg}\%$ calculated before the last (current) generation included in the current $SMA\_Avg\_Expr_{curr}$.

Thus, the DNSP changes the size of subpopulations based on the exploration status, as follows:

1. If ($SMA\_Avg\_Expr_{curr} < SMA\_Avg\_Expr_{previous}$), the size of each subpopulation is randomly increased considering the integer interval $[M+1, Max\_NS_{node}]$. Let M be the current number of individuals in the subpopulation, the new population size is given by $M_{new} = rand(M+1, Max\_NS_{node})$, where $rand$ generates an

integer number in the range $[M+1, Max\_NS_{node}]$. This condition points out that the current SMA of $Expr_{avg}\%$ is less than the previous SMA, i.e., the exploration is decreasing according to the compared periods. Then, increasing the number of solutions may increase the exploration in terms of population diversity. When this condition is satisfied, the procedure randomly initializes $M_{new} - M$ individuals for each niche in order to meet the new subpopulation size;

2. Contrarily, if ($SMA\_Avg\_Expr_{curr} > SMA\_Avg\_Expr_{previous}$), the size of each subpopulation is randomly decreased considering the integer interval $[Min\_NS_{node}, M-1]$. The new population size is given by $M_{new} = rand(Min\_NS_{node}, M-1)$. Such a condition indicates that the current SMA of $Expr_{avg}\%$ is higher than the previous SMA, i.e., the exploration is increasing. Reducing the subpopulation size may decrease the exploration and increase the exploitation in terms of population diversity. When this condition is satisfied, the procedure discards the worst $(M - M_{new}) \times N_{nodes}$ individuals of the entire population to meet the new subpopulation size.

We note that the new subpopulation size is applied to all nodes of the tree as the procedure considers the SMA of the $Expr_{avg}\%$ values of the entire MA population.

As already mentioned, the conditions above to change the subpopulation size are only applied when the LS restarting and the performance verification procedures are executed, which indicates that the algorithm cannot improve the best solutions for a given number of generations. Thus, we tried to relate this stagnation with the SMA of Expr/Expt efforts of the algorithm. If the exploration decreases in a given period, the subpopulation size is increased to overcome local minima. Alternatively, if the exploration is increasing, the subpopulation size is reduced, increasing exploitation, to enforce the convergence and focus the search on the best solutions of each node.

Moreover, most of the works presented in Section 4 that adopt a dynamic multi-population size strategy randomly determine the new subpopulation size based on the quality of the best solution. The random definition of the niche size does not consider the current status of the search process, that is, the population diversity degree, Expr/Expt efforts, and the algorithm's performance to point out the appropriate direction on the change in the population. With this, the DNSP aims to define which direction the search process should follow before defining the new size of subpopulations.

Third scenario: Whenever the niche restarting procedure is executed, the DNSP is performed. However, the niche size is not changed as in the above scenarios. In this case, the DNSP only executes the clustering-based Speciation algorithm to reorganize the solutions into the nodes according to their similarities.

Reorganization of the subpopulations: After performing any of the above scenarios, the clustering-based Speciation (Algorithm 2) is applied to reorganize the individuals into the nodes according to their similarities. Thus, the niching algorithm receives as input parameter the current number of nodes in the tree ($NCl = N_{nodes}$) and the number of individuals in the entire population considering the new subpopulation size ($NS = M_{new} \times N_{nodes}$), when changed. The DNSP is described in Algorithm 6, which receives as input parameters the subpopulations of nodes $nodes$, the exploration value $Expr\%$ of each one and the period in terms of generations n_gen used to calculate the SMA.

5.2.10 Priority Order of the Components

To avoid search strategy overlap regarding the procedures for control of diversity/convergence and performance, we defined priority order of execution per generation for each one. Thus, the TRP, the LS restarting, the niche restarting, the performance verification, and the DNSP procedures are performed based on the following priority orders:

1. The TRP is performed whenever its application condition is satisfied (Section 5.2.7);
2. The LS restarting is performed whether its application condition is satisfied and the TRP is not performed (Section 5.2.6);
3. The niche restarting is applied whether its application condition is met and the LS restarting is not executed (Section 5.2.8);
4. The performance verification procedure is executed whether its application condition is satisfied and none of the above components are performed (Section 5.2.8);
5. The DNSP is performed whenever any of the above components is applied (Section 5.2.9).

Such order of execution was adopted to prevent the overlapping of search strategies and especially not to bias the search process by applying different procedures with a similar purpose at the same generation. Figure 5.3 illustrates the priority orders described above.

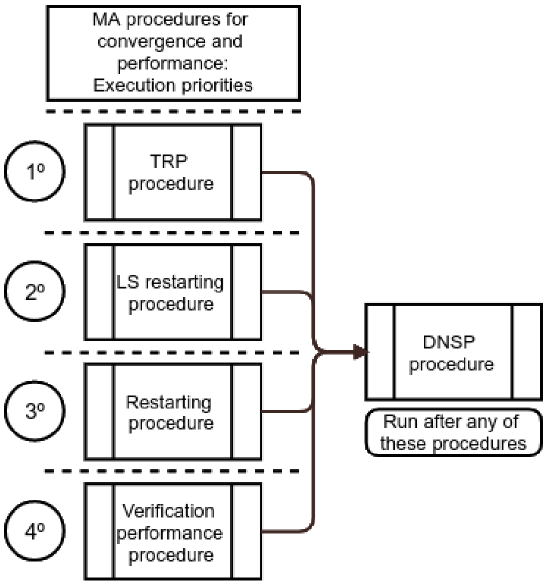
Algorithm 6 Pseudocode of the dynamic niche size procedure.

```
Require: $nodes$: node subpopulations; $Expr\%$: exploration values of $nodes$; $n\_gen$: number of
    generations used to calculate the SMA
Ensure: $nodes$: node subpopulations after the DNSP
 1: $Min\_NS_{node} = 5$
 2: $Max\_NS_{node} = Max\_NS/N_{nodes}$
    //Exploration inference of the entire population
 3: calculate $Expr_{avg}$ from average of $Expr_i\%$ values of each $node_i$ in $nodes$ (Equation 5.10)
 4: store $Expr_{avg}$ to be considered in the SMA of $Expr_{avg}\%$ values
 5: $SMA\_Avg\_Expr_{previous} \leftarrow SMA\_Avg\_Expr_{curr}$
 6: $SMA\_Avg\_Expr_{curr} \leftarrow$ calculate SMA of $Expr_{avg}\%$ values through $n\_gen$ generations
 7: if ($TRP$) was performed then //First scenario
 8:     $M_{new} \leftarrow rand(Min\_NS_{node}, Max\_NS_{node})$ //Increase/decrease the subpopulation size
 9:     initialize or discard solutions for each $node_i$ in $nodes$ depending on the $M_{new}$
10: end if
11: if ($LS\_restarting$) or ($PerformancVerification$) were performed then //Second scenario
12:     if $SMA\_Avg\_Expr_{curr} < SMA\_Avg\_Expr_{previous}$ then
          //Exploration is decreasing
13:         $M_{new} \leftarrow rand(M + 1, Max\_NS_{node})$ //Increase the subpopulation size
14:         randomly initialize the $M_{new} - M$ solutions for each $node_i$ in $nodes$
15:     else
          //Exploration is increasing
16:         $M_{new} \leftarrow rand(Min\_NS_{node}, M - 1)$ //Decrease the subpopulation size
17:         discard the worst $M - M_{new}$ solutions of each $node_i$ in $nodes$
18:     end if
19:     restart the $Expr_{avg}\%$ values stored
20: end if
21: if ($TRP$) or ($LS\_restarting$) or ($niche\_restarting$) or ($PerformancVerification$) were per-
    formed then //All scenarios
      //Apply the niching strategy using the updated niche size $M$
22:     $nodes \leftarrow ClusteringSpeciation(nodes, M_{new} \times N_{nodes}, N_{nodes})$
23: end if
24: return $nodes$
```

5.2.11 Summarization of the Framework-based MA

The framework-based MA was proposed for continuous optimization but focused on global optimization with multimodal objective functions. It aims to be a general optimization method that can be easily adapted to other search strategies and components. Table 5.1 summarizes the main components incorporated in the framework, the search strategies, and concepts related to them. The Algorithm 7 shows the pseudocode of the MA, which receives as input parameters the initial number of layers N_{layers} in the tree, the maximum number of individuals Max_NS in the entire population of the MA, the maximum number of fitness function evaluations Max_Evls, the number of generations g_{core}

Figure 5.3: Exemplification of the priority order of execution for each procedure


MA procedures for convergence and performance: Execution priorities
1º
TRP procedure
2º
LS restarting procedure
3º
Restarting procedure
4º
Verification performance procedure
DNSP procedure
Run after any of these procedures

Source: From the author (2022).

executed by the core metaheuristic and the period in terms of generations n_gen used to calculate the SMAs. It is noted that the solutions of subpopulations are kept sorted from best to worst according to the fitness values throughout the algorithm's execution. The following section describes the adopted core metaheuristic and the modified versions to deal with multimodal optimization. We observe that the parameter setting and the intermediate variants used to design this version of the method are further detailed in the section of computational experiments (Section 6.2).

Table 5.1: Summarization of the MA components, search strategies and main concepts related to them

Component	Search strategy	Search concept
`Tree initialization`	Clustering-based Speciation	Niching
`Core search algorithm`	Any metaheuristic	Niche optimization
`Inner recombination`	Uniform crossover with DC	Intra-niche exploration and diversity preservation
`Node interactions`	Uniform crossover with RTS	Inter-niche exploration and diversity preservation
`LIP`	LS chain with SW algorithm	Global/local balance; LS with self-adapting param.
`LS restarting`	Restarting of subpopulation	Population diversification
`TRP`	Linear reduction of tree nodes	Balance between Expr/Expt;
`Control mechanism`	Restarting and performance control	Population diversification; Expr/Expt balance
`DNSP`	Adaptive niche size; Re-clustering	Expr/Expt balance; overcome search issues

Source: From the author (2022).

5.2.12 Core Metaheuristic

The proposed framework-based MA was combined with a modified version of the ABC algorithm (KARABOGA; BASTURK, 2007) employed as the core metaheuristic responsible for the MA subpopulations' inner exploration. Each node of the tree incorporates an independent execution of the algorithm. This metaheuristic is based on the foraging behavior of a bee colony, which is focused on multivariate numerical function optimization (KARABOGA; BASTURK, 2007; AKAY; KARABOGA, 2012). ABC becomes a popular optimizer because it presents few control parameters, simple structure, ease of implementation, and robust optimization performance when compared with other population-based metaheuristics (KARABOGA; BASTURK, 2008; KARABOGA; AKAY, 2009; GAO; LIU; HUANG, 2012; KARABOGA et al., 2014; DOKEROGLU et al., 2019). The standard ABC and some relevant variants were previously discussed in Sections 4.5.1 and 4.5.2. Also, it should be noted that other ABC variants were already explored in our previously published works, as shown in (CORRÊA; DORN, 2018; CORRÊA; DORN, 2019; CORRÊA; DORN, 2020).

Thus, according to the literature, we believe that the synergy of concepts between the MA and ABC algorithms seems suitable to the optimization domain understudy in the sense of efficient search space exploration and refinement of solutions. The core search algorithm was designed as a hybrid algorithm that uses a self-adaptive strategy for its control parameters, population division based on the solutions' quality, and DE-based mutation to further enhance the search abilities of the ABC.

Hybrid Self-adaptive ABC

As mentioned in Sections 4.5.1 and 4.5.2, the ABC (Algorithm 1) has the ability of search space exploration but presents some inefficiencies of exploitation, which can imply in slow convergence rate and stagnation (ZHU; KWONG, 2010; AKAY; KARABOGA, 2012; LI; NIU; XIAO, 2012). One of the possible reasons for such an issue may be that the mutation search strategy used in the standard ABC (Equation 4.1), when a candidate solution is generated, randomly updates only one objective variable of the corresponding parent solution at a time, which makes the disturbance in variables small. Moreover, the standard Equation 4.1 is entirely random in terms of direction, and it may also not consider the information of the fittest solutions of the population. Hence, the promising evolution direction is not fully exploited at all. The mutation focuses more on exploration and relatively neglects exploitation (KIRAN; FINDIK, 2015; CUI et al., 2016; GAO et al., 2018).

According to the literature, several ABC variants have been proposed over the years by mainly modifying the solution search equation and combining the ABC with other search strategies (CAI et al., 2020). With this, we designed a modified version of the ABC in an attempt to overcome such issues and accelerate the algorithm's convergence inside each node of the tree.

Population division strategy: Therefore, each ABC subpopulation is internally subdivided into two groups according to the solutions' fitness quality. Each group employs a distinct mutation search equation to enforce the exploration from part of the population and enable the convergence to promising regions from the other part. The population division strategy is adopted to balance the Expr/Expt mechanisms of the ABC by avoiding the use of a single mutation for solutions in different stages of convergence. So it tries to employ the most reasonable search equation for each group regarding the optimization stage of the individuals.

This strategy splits the individuals of a given node into two levels concerning their fitness values: (*i*) the superior individuals, which have better fitness values and are suitable for exploitation; and (*ii*) the inferior individuals, which represent the poorer solutions and are suitable for exploration. As fitness division threshold used to classify the solutions, we adopted the midpoint of the set of fitness values (md), given as follows:

$$md = \frac{f(X_{worst}) + f(X_{best})}{2} \tag{5.11}$$

Where $f(X_{worst})$ and $f(X_{best})$ are the objective values of the worst and best solutions of the current subpopulation. Each subpopulation has its md based on its set of solutions. The midpoint of the set of fitness values does not consider the number of solutions allocated in each group and can be easily calculated. Then, individuals of a given niche with similar fitness may focus on exploration or exploitation and not on both based on the split of the population. The size of the groups varies according to the optimization status of the subpopulation.

When the core algorithm is applied to a given node, at the beginning of each generation, each individual of the subpopulation is compared with the division threshold md, where the solutions with fitness values better than the threshold are categorized as superiors. In contrast, individuals with fitness worse than the midpoint are classified as inferior individuals.

The superior individuals are guided by the best individual of the subpopulation to move toward the global of this niche and hence accelerate the population convergence. These individuals are mutated by the gbest search equation of the GABC (ZHU; KWONG, 2010). On the contrary, inferior solutions should explore other regions of the search space and preserve population diversity. The inferiors are updated via the solution search equation “DE/rand/1” from the DE algorithm, described by Equation 4.4 in Section 4.5.3. “DE/rand/1” is a random mutation with no bias to any search direction, which favors the search space exploration.Thus, the population division strategy is used to not only accelerate the convergence of the best solutions but also to maintain some degree of diversity in order to overcome local optima and premature convergence.

The proposed ABC follows the same stages of optimization of the standard ABC described in Algorithm 1 (Section 4.5.1). It consists of three steps that simulate the foraging task performed by specialized bees of a honeybee. The employed bees’ stage is responsible for updating all solutions of the subpopulation through a mutation search equation. The onlooker bees’ stage selects to update the fittest solutions based on their fitness values via the ranking-based selection. The same updating process of the employed bees’ step is applied to the selected solutions. Moreover, the scout bees’ stage is used to diversify the population, discarding the most inactive individual and generating a new one. The most inactive solution is the one that does not suffer improvements for a given number of generations. In this sense, we observe that the division of population and classification of individuals as superiors/inferiors are done at the beginning of each algorithm generation. Such split determines which solution search equation is employed to mutate a given individual over the ABC stages.

Superior individuals: In employed and onlooker bees’ stages, the superior individuals are updated by the gbest search equation presented in the GABC (ZHU; KWONG, 2010), which was proposed to enhance the exploitation ability of the ABC. The update operation in both stages of optimization generates a new solution υ_i from the i-th existing solution X_i of the subpopulation, such that X_i is a superior individual. This operation is done by the gbest equation (Equation 4.2), previously described in Section 4.5.2, but shown again as following:

$$\upsilon_{ij} = x_{ij} + \delta_{ij}(x_{ij} - x_{kj}) + \gamma_{ij}(gbest_j - x_{ij}) \tag{4.2}$$

Where $i = 1, \cdots, SN$, and $j = 1, \cdots, D$. SN denotes the number of solutions in the subpopulation and D is the problem dimension. x_{ij} is the j-th variable of X_i, υ_{ij} is the value assigned to x_{ij}, and x_{kj} represents the j-th variable of the k-th solution ($k = 1, \cdots, SN$), randomly selected among all solutions of the subpopulation, such that $k \neq i$. δ_{ij} is the scaling factor, which is a random real number in the range $[-1, 1]$, $gbest_j$ is the j-th position of the current best individual of the niche, and γ_{ij} is a random real number in the range $[0, 1.5]$. The number of mutated variables in each update operation is controlled by the rate-MR parameter, further explained. υ_i replaces X_i if it is better regarding their fitness function values.

The gbest search equation considers the influence of the fittest solution of the niche and its neighborhood to guide the mutation operation towards promising directions and then increase the convergence rate, refining the accuracy of individuals. As the superior individuals have better fitness than the inferiors, gbest mutation is used to preserve their quality and improve the local search mechanism of the algorithm. From Equation 4.2, it is observable that the candidate solution X_i, the randomly selected solution X_k, and the current gbest solution $gbest$ collaborate to determine the step size and direction of the variables of the newly generated solution υ_i. Thus, the solutions generated from gbest mutation tend to search toward a potential global optimum.

Inferior individuals:

Inferior individuals are solutions with fitness values worse than or equal to the subpopulation division threshold md, which indicates that these points have not yet reached a considerable degree of convergence and could be suitable for exploration. As gbest search equation (Equation 4.2) for superior individuals can enforce the ABC exploitation, it also tends to get trapped at local points more often than random mutations due to the homogeneous orientation direction. So this low level is used in attempt to maintain some diversity degree intra-niche as a counter-balance regarding the superior level.

To do so, we adopted the "DE/rand/1" search equation from the DE algorithm, previously described by Equation 4.4 in Section 4.5.3. Similar to the update operation in ABC, in the employed and onlooker bee updating steps, a new solution υ_i is generated through the "DE/rand/1" from the i-th existing solution X_i of the subpopulation, such that X_i is an inferior individual. The equation randomly selects two individuals of the subpopulation, and their difference multiplied by a scale factor $F \in [0, 1]$ is added to the third randomly selected individual. This operation is illustrated again as follows:

$$\upsilon_i = X_{r1} + F(X_{r2} - X_{r3}) \tag{4.4}$$

Where the indexes $r1$, $r2$, and $r3$ are randomly chosen within a set $[1, SN]$, such that SN

is the number of solutions in the subpopulation and $r1 \neq r2 \neq r3 \neq i$. We note that the number of mutated variables in each update operation is also controlled by the rate-MR parameter, further described. The update ends with a greedy selection between the new solution and the former regarding their fitness function values.

The "DE/rand/1" is a random search equation that lies in the fact that it has no bias to any search direction, such as the oscillation phenomenon (CUI et al., 2016; CUI et al., 2017). Besides, "DE/rand/1" tends to present a more random behavior focused on diversification than the standard ABC equation (Equation 4.1), due to the number of randomly selected solutions and organization of the mutation terms. With this, through this level, the inferior individuals are guided by differential operations between randomly selected solutions in order to preserve their exploratory profile and enhance the search space exploration, thus supporting the niche to avoid local optima and premature convergence. Finally, this hybridization of the ABC and DE algorithm should provide a more efficient method than using a single metaheuristic (JADON et al., 2017).

Mutation rate-MR strategy: In the standard ABC and GABC variant, the employed and onlooker bees generate new solutions by changing only one dimension of their parent solutions, which can be one of the reasons for the algorithm's slow convergence. With this, as mentioned in Section 4.5.2, the mutation rate-MR strategy (AKAY; KARABOGA, 2012) was proposed to mutate more than one variable at a time in the ABC update operation. This strategy is very similar to the binomial crossover in the DE algorithm, earlier described by Equation 4.5. In the rate-MR, each variable j of the solution X_i is probabilistically updated according to the control parameter MR. Thus, a variable is updated with a probability of $MR\%$ whether $rand(0, 1) < MR$, where $rand$ generates a real number in the range $[-1, 1]$.

Therefore, we incorporated the rate-MR in the ABC update of solutions for both superior and inferior levels to regulate the number of mutated variables in each operation. This strategy is used to increase the disturbance in the candidate solutions' variables and promote exploitation through the gbest search equation for superior solutions and keep population diversity by the "DE/rand/1" equation for inferiors. Then, the employed and onlooker bees' phases consider more dimensions to be mutated at each operation based on the defined threshold MR. In this work, we used rate-MR with $MR = 0.4$ by the suggestion of Akay and Karaboga (AKAY; KARABOGA, 2012) for continuous optimization. The lower value for MR may imply the slow improvement of solutions, while the higher value for MR can cause an unnecessary diversification in the population. Therefore, an objective variable is updated with a probability of 40%, if $rand(0,1) < 0.4$.

ABC control parameters: It is known that the performance of any metaheuristic is based on its search components and control parameters. The mechanisms adopted in our ABC version aims to balance the Expr/Expt efforts inside each node of the tree with the support of the MA framework. The control parameters in the proposed ABC encompass the population size SN, the scaling factors δ and γ for gbest search equation and the scaling F for "DE/rand/1", the mutation rate MR, and the "limit" parameter l for scout bees. The MA controls the size of subpopulations submitted to the ABC, and the MR parameter was defined above. Also, we adopted the common proportion $l = SN \times D$ for the threshold of discarding the scout bees, following the suggestion of Akay and Karaboga (AKAY; KARABOGA, 2012) as a good setting. SN is the population size, and D represents the problem dimension. Notably, this proportion $l = SN \times D$ seems to be especially reasonable in our case due to the dynamic size of the subpopulations throughout the search, regulated by the control search strategies incorporated in the MA for convergence and performance.

Finally, the search equations' scaling factors δ, γ and F are self-adapting control parameters throughout the optimization process. This self-adaptive mechanism was inspired in the jDE algorithm (BREST et al., 2006; BREST; MAUČEC; BOŠKOVIĆ, 2019), previously explained in Section 4.5.3. Each individual X_i of the subpopulation has its control parameter values δ_i, γ_i and F_i, which are evolved together with the individual throughout the process. The concept of self-adaptation is that better solutions tend to survive to the next generation and propagate their variables along with the parameter set responsible for high quality (ALETI; MOSER, 2016). With this, different from the standard ABC and GABC, the control parameters δ and γ in our method are kept the same for a given candidate solution X_i during the update operation, which yields δ_i and γ_i for X_i. For instance, in the standard ABC, a random value of $\delta \in [0,1]$ is assigned to each mutant variable of X_i, i.e., δ_{ij} for the j-th position of X_i. This modification was done due to the self-adaptive strategy adopted for the scaling parameters. New parameter values for $\delta_{i,gen+1}$, $\gamma_{i,gen+1}$ and $F_{i,gen+1}$ are defined before the update operation of X_i, as shown in the following equations:

$$\delta_{i,gen+1} = \begin{cases} \delta_l + rand_1 \cdot \delta_u & \text{if } (rand_2 < \tau) \\ \delta_i & \text{otherwise} \end{cases} \tag{5.12}$$

$$\gamma_{i,gen+1} = \begin{cases} \gamma_l + rand_3 \cdot \gamma_u & \text{if } (rand_4 < \tau) \\ \gamma_i & \text{otherwise} \end{cases} \tag{5.13}$$

$$F_{i,gen+1} = \begin{cases} F_l + rand_5 \cdot F_u & \text{if } (rand_6 < \tau) \\ F_i & \text{otherwise} \end{cases} \tag{5.14}$$

Where $i, gen+1$ represent the definition of the control parameter of the candidate solution X_i that will be used in the next update operation. $rand_j$ ($j \in 1, 2, 3, 4, 5, 6$) are random values in the range $[0, 1]$, whereas the τ value is the probability to modify the scaling factors δ, γ and F. The control parameters are initialized at the beginning of the MA execution in the initialization step of the framework.

According to the jDE algorithm (BREST et al., 2006), we adopted for τ, F_l and F_u, the fixed values 0.1, 0.1, 1.1, respectively (BREST; MAUČEC; BOŠKOVIĆ, 2019). Also, according to the GABC equation (Equation 4.2), δ_l, δ_u, γ_l and γ_u are set to -1, 2, 0 and 1.5, respectively. With this, the δ parameter can get values from $[-1, 1]$, γ can assume values from $[0, 1.5]$ and F can take values from $[0.1, 1.2]$. Lastly, it is observed that $\delta_{i,gen+1}$, $\gamma_{i,gen+1}$ and $F_{i,gen+1}$ are calculated before the mutation is performed. So they influence the update and selection operations of the newly generated solution v_i.

The complete ABC algorithm: The hybrid ABC algorithm designed in this work is used as the core metaheuristic for niche exploration in the MA framework. As already described in Section 5.2.3, each node of the tree runs the core algorithm independently to optimize its subpopulation. The Algorithm 8 describes the pseudocode of the hybrid ABC. The method is executed by g_{core} generations in each node and receives as input parameter the subpopulation of a given node to be optimized, and the number of generations (g_{core}) to be executed as stop criterion.

Briefly, the main loop of the algorithm represents the evolutionary process, which is iterated for g_{core} generations. In the employed bees' stage (Algorithm 8, line 4), all solutions are updated according to the population division strategy (Algorithm 8, line 3), where the superior individuals are mutated by the gbest search equation (Equation 4.2) (Algorithm 8, line 9) and the inferior ones are updated by the mutation operation "DE/rand/1" (Equation 4.4) (Algorithm 8, line 19). The scaling factor parameters δ, γ and F for these equations are initialized at the beginning of the MA execution in the initialization step of the framework and self-adapted during the ABC execution in the employed and onlooker bees' stages (Algorithm 8, lines 13 and 23, respectively). So the ABC encodes the scaling factors alongside the objective variables in a single vector, which are evolved together during the optimization process. After the definition of δ_i and γ_i for pop_i, the gbest search equation (Equation 4.2) is applied to v_{ij} (Algorithm 8, line 9).

In the onlooker bees' stage (Algorithm 8, line 28), solutions are selected based on their fitness values via the ranking-based selection (Algorithm 8, line 29). The selected solutions are mutated following the same update procedure as the previous step.

We observe that the difference between the first two steps is that in the first stage (employed bees), all population solutions are updated. In the second stage (onlooker bees), only the probabilistically selected solutions are updated, giving to the fittest ones more chances to be mutated.

In the scout bees' stage (Algorithm 8, line 36), a solution that has not been improved for a number l of generations is discarded (Algorithm 1, line 38), and a new one is inserted into the population (Algorithm 8, line 39). We observe that each node also performs g_{core} inner recombination operations in an interspersed way with the core algorithm, as previously explained in Section 5.2.3. Thus, at the end of each generation of the ABC, the recombination is applied once (Algorithm 8, line 41). Finally, the algorithm returns the optimized population after the stop criterion is satisfied.

5.3 Proposed Method for Multimodal Optimization

Focusing on multimodal optimization, we incremented the previously described MA framework based on the issues concerned with discovering and maintaining the existing global optima of a given objective function. As presented above, the tree data structure of the method was adopted as a niching strategy, splitting the population into distinct subpopulations. This strategy was designed to overcome the multimodality issues over the search space and support the population diversity/convergence concerning distinct optimization stages. The proposed MA already encompasses independent intra-niche optimizations and interactions between niches, as well as procedures for local improvements and control of convergence and performance. Besides, some hyper-parameters are dynamically adjusted throughout the optimization process to ease the search space optimization according to the degree of Expr/Expt efforts. Such strategies for performance control are widely used to prevent premature convergence to local optima and slow the convergence rate when necessary (GLIBOVETS; GULAYEVA, 2013; LI et al., 2016;

SER et al., 2019).

Nonetheless, as mentioned earlier in Sections 2.3 and 4.6, stochastic metaheuristics tend to naturally converge to a single global optimum due to the genetic drift inherent to the evolutionary process (BELDA et al., 2007). With this, the effective discovery and preservation of multiple distinct solutions throughout the algorithm's execution become essential when dealing with multimodal optimization with multiple global optima (WANG et al., 2019). As stated by Li et al. (LI et al., 2016), the evolutionary population does not have to totally converge to single solutions, each corresponding to a global optimum. In this sense, the strategies for controlling the method convergence and performance, previously presented, attempt to reach some equilibrium state, where solutions would keep oscillating around a stable optimum but without reaching complete convergence.

Therefore, due to the designed MA search structures, the algorithm has not been so changed for multimodal optimization. The main concern consists of keeping the found optimal points stably and not losing them while locating new ones throughout the search process. So a simple external archive was incorporated to store the identified global optima when converged regions of the search space are detected. By using the MA restarting strategies to control such converged niches, whenever a discard of solutions occurs, the individuals are stored into the archive to keep the found promising solutions separate from the current population. In contrast, the framework generates new solutions to continue exploring new areas. In our approach, the discard of solutions is done when the procedures of LS restarting (Section 5.2.6), niche restarting (Section 5.2.8) or scout bees in the ABC core metaheuristic (Section 5.2.12) are applied. The archived individuals are kept in the archive until the end of the search and considered to compute the final optimization results.

Moreover, some of the control parameters concerning the implemented framework components were adjusted to better fit the benchmark test functions for multimodal optimization and its limited budget. We note that the parameter setting used for this variant of the method is further detailed in the section of computational experiments (Section 6.3).

Finally, the proposed method for the PSP problem is described in the following section. As already explained, it was designed via an incremental approach encompassing almost all the search strategies incorporated in the MA framework and detailed so far.

5.4 Proposed Method for the 3-D PSP Problem

To deal with the PSP problem as a multimodal problem, we incremented the framework-based MA, presented in the sections above (Sections 5.2 and 5.3), taking into account the problem-specificities and its optimization challenges. In addition, we used as the baseline for the development of the method detailed in this section, our previously published method for the PSP in Corrêa and Dorn (CORRÊA; DORN, 2020).

With this, the method presented in this work can be divided into two main stages: (*i*) sampling of structural models and initialization (Section 5.4.2); and (*ii*) optimization of the structures from the previous step via the MA framework (Section 5.4.3). We tried to incorporate in the framework for the PSP the problem-dependencies and data knowledge from experimentally determined 3-D protein structures to turn it more robust to the problem under study.

5.4.1 Conformational Preferences of Amino Acids

The Angle Probability List (APL) aims to determine the amino acid (aa) conformational preferences of a given target protein (CORRÊA et al., 2016; BORGUESAN; INOSTROZA-PONTA; DORN, 2016; CORRÊA; DORN, 2020). It determines the aa preferences by analyzing previous occurrences in proteins whose structures were experimentally determined. In the works of Corrêa et al. (CORRÊA et al., 2016; CORRÊA; DORN, 2020), we demonstrated the use of APL incorporated in a MA for the PSP problem. It was employed to define the aa angular preferences considering the influence of its neighbors in the protein sequence. Besides, Borguesan et al. (BORGUESAN; INOSTROZA-PONTA; DORN, 2016) developed a web server, called NIAS[1] (Neighbors Influence of Amino acids and Secondary structures). NIAS aims to provide the community with a tool that allows the information extraction about conformational preferences of amino acids over the generation of different APL types.

The APL strategy used in this work was presented by Corrêa and Dorn (CORRÊA; DORN, 2020) from protein structures of the PDB. According to the authors, a set of filters were applied to guarantee the experimental data quality. A set of 11,130 protein structures were selected, and all of them were determined experimentally by X-ray crystallography. Only mature structures with a resolution less than or equal to 2.5 and deposited in the

[1]<sbcb.inf.ufrgs.br/nias/>

PDB until December 2014 were used for the APL construction. The resolution index evaluates the detail level of X-ray diffraction data, which is considered a good indicator of the experimental structure quality (HOVMÖLLER; ZHOU; OHLSON, 2002). Another structural quality index used in the protein structures filtering was the R-observed. It was used to evaluate the similarity between the crystallographic model and the X-ray diffraction data. All structures with R-observed above 0.20 were removed from the set. Only one of the homologous protein structures found with sequences identity at most 30% was retained to avoid redundancies. From this set, the authors considered amino acids well-defined position with B-factor threshold less than or equal to 30^2, and Occupancy equal to 1. At the end of the filtering process, a total of 2,399,401 amino acids were generated for further analysis. The software STRIDE[2] (HEINIG; FRISHMAN, 2004) was used to assign the aa secondary structures in experimental structures.

The authors designed different APL types to incorporate structural information from the experimental protein set into the method (CORRÊA et al., 2016; BORGUESAN; INOSTROZA-PONTA; DORN, 2016). They have constructed a database to represent the aa conformational preferences, based on experimental information filtered from the PDB, as described above. For each aa type, occurrences of the dihedral angles ϕ and ψ were computed according to the experimental structures in the filtered set. Also, the aa conformational preferences were computed regarding their respective SS. Corrêa and Dorn (CORRÊA; DORN, 2020) expanded the proposed APL versions of Borguesan et al. (BORGUESAN et al., 2015; BORGUESAN; INOSTROZA-PONTA; DORN, 2016) and Corrêa et al. (CORRÊA et al., 2016) to further exploit the conformational preferences with more specialized structural information.

Therefore, the APL considers the angular occurrences of a given aa and its SS, known as reference amino acid (aa_{ref}), and the influence of the neighboring amino acids to define its conformational preferences. Thus, such a technique assigns the dihedral angles to the target amino acids by analyzing the conformational preferences of such amino acids in experimental structures regarding the SS and the neighborhood of amino acids. The APL encompasses four distinct types of combination concerning the aa_{ref}:

1. APL-1 which considers only the aa_{ref} and its SS;
2. APL-2l that considers the influence of the aa at the left and its respective SS;
3. APL-2r that considers the influence of the aa at the right and its SS;
4. APL-3 which considers the influence of the amino acids at the left and right and

[2]<webclu.bio.wzw.tum.de/stride/>

their SS.

This set of APL combinations is referenced as APL-neighborhood.

Also, another APL variation, known as APL-centroid, was proposed in Borguesan et al. (BORGUESAN; INOSTROZA-PONTA; DORN, 2016), and Corrêa and Dorn (CORRÊA; DORN, 2020). It considers only the aa_{ref} and its SS, ignoring the neighborhood of amino acids, but still considering the SS of them. Thus, to determine the aa_{ref} conformational preferences from the APL-centroid, only its SS and the SS of the neighboring amino acids are used without considering their types, thus accepting any aa type with the matching SS. The APL-centroid can be divided into three types depending on the number of amino acids considered in the generation of conformational preferences:

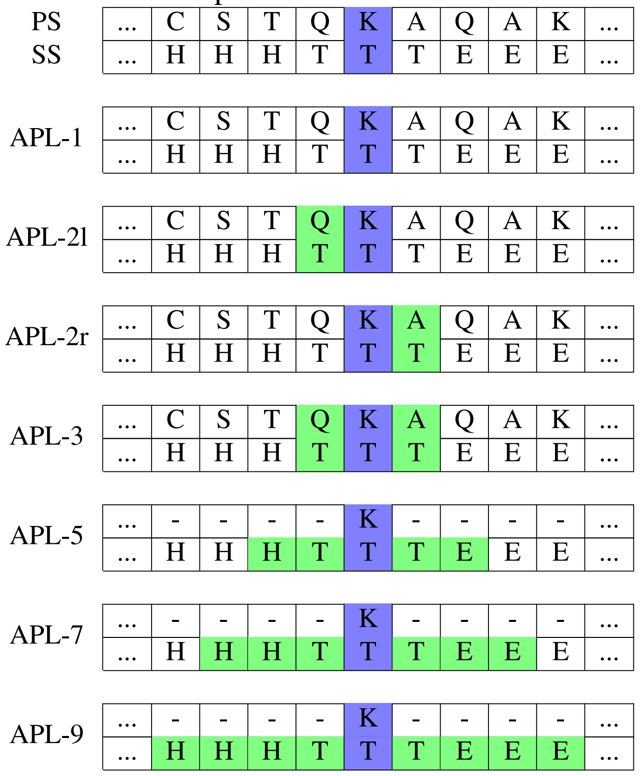
1. APL-5 which considers the SS influence of the two amino acids at the left and the two amino acids at the right of the aa_{ref};
2. APL-7 which considers the SS influence of the three amino acids at the left and the three amino acids at the right;
3. APL-9 that considers the SS influence of the four amino acids at the left and the four amino acids at the right over the aa_{ref}.

According to the authors, this APL variation set, APL-centroid, was required due to the small amount of experimental data present in some aa and SS combinations of APL-3, thus providing a greater variety of conformational data to users.

Figure 5.4 illustrates the different APL types through the aa sequence "`CSTQKAQAK`" with SS "`HHHTTTEEE`" taken from a segment of the 1ACW protein (PDB ID). In this exemplification, the aa chosen as reference (aa_{ref}) was `K` (lysine) with SS of loop (`T`), both shown in blue. Neighboring amino acids that influence the conformational preferences of aa_{ref} are highlighted in green according to the APL type. We note that unlike the fragment assembly approaches, as previously seen in the method of Rosetta (Section 4.7.1), in the APL, each aa combination is used to assign the angles only to the aa_{ref}, whereas in fragment assembly, the angles of all the amino acids that encompass the fragment are assigned.

To deal with the PDB information, the authors have built discretized histograms ($H_{aa,z}$) of $[-180°, 180°] \times [-180°, 180°]$ cells, where $\{H_{aa,z}(i, j) = x | x \in \mathbb{N}\}$ and x denotes the number of times (occurrences) that an aa (or combination of amino acids) with SS z has presented a pair of dihedral angles (ϕ and ψ) of values i and j, respectively, in the experimentally determined structural set. We note that the database's dihedral an-

Figure 5.4: Schematic representation of the different APL types for the aa sequence "`CSTQKAQAK`" with SS "`HHHTTTEEE`", removed from a segment of the 1ACW protein (PDB ID). The blue cells denote the aa_{ref} and its respective SS, and the green ones, depending on the APL type, highlight the neighboring amino acids considered in the definition of the aa_{ref} conformational preferences


PS ... C S T Q K A Q A K ...
SS ... H H H T T T E E E ...
APL-1
APL-2l
APL-2r
APL-3
APL-5
APL-7
APL-9
... - - - - K - - - - ...

Source: From Corrêa and Dorn (CORRÊA; DORN, 2020).

gles, which gave rise to the APLs, had their values rounded to fit into the histograms' discrete space. Each $H_{aa,z}$ represents a different combination between amino acids and secondary structures that count for the definition of the aa_{ref} conformational preferences. The number of amino acids considered in each combination varies according to the APL type.

Regarding the APL-neighborhood set, different combinations of amino acids and secondary structures of up to three amino acids (1-3 aa) were generated, considering the aa_{ref} neighborhood for combinations of length greater than 1 aa. For the APL-centroid set, different combinations involving the secondary structures of the aa_{ref} neighborhood were established to define the aa_{ref} conformational preferences, forming combinations with SS lengths of 5 (APL-5), 7 (APL-7) and 9 (APL-9) amino acids.

Therefore, each cell (i, j) of the histogram $H_{aa,z}$ counts the number of times an aa (or combination of amino acids) has a dihedral angle pair (i$\leq$ ϕ <i+1, j$\leq$ ψ <j+1) corresponding to the SS z in the experimental database. For each cell (i, j) of a given

histogram, the sum value of the eight neighboring cells was also added to highlight the most abundant conformational regions, according to the smoothing Equation 5.16.

$$APL_{aa,z}(i,j) = \frac{H'_{aa,z}(i,j)}{\sum\limits_{x,y} H'_{aa,z}(x,y)} \tag{5.15}$$

$$H'_{aa,z}(i,j) = \sum_{r=i-1}^{i+1} \sum_{s=j-1}^{j+1} H_{aa,z}(r,s) \tag{5.16}$$

Where r and s represent the positions (i,j) of the eight neighbors of a given cell in the histogram matrix $H_{aa,z}(r,s)$. Then, for each histogram $H'_{aa,z}$, the Angle Probability List (APL$_{aa,z}$) is calculated (Equation 5.15), denoting the frequency of each cell.

In this work, all APL types (neighborhood and centroid) were used in the solution initialization step of the optimization method by generating different amino acid combinations (length of 1-3 aa and 5-9 aa) in an attempt to feed the algorithm with high-quality solutions when compared with those randomly initialized. Besides, APL-1 was used to constraint the search space when applying mutation operators, such as in the ABC algorithm, to avoid non-existent angular positions in APLs.

5.4.2 Sampling of Protein Structures and Solution Initialization

The first step of the framework for the PSP is responsible for performing the sampling of protein models. They are generated from the APL (Section 5.4.1), considering the aa probability of occurrence based on conformational preferences of previously known protein structures. In this step, generated individuals are submitted to filtering and clustering processes to overcome the search space roughness and provide feasible initial solutions to the optimization step. The sampling process of individuals refers to the solution generation according to the primary sequence of the target protein. An individual is a vector of real values (objective variables) representing dihedral angles, which describes the target protein's chains, characterizing the structural model's computational representation through the dihedral angles set formed from the target primary sequence, described in Chapter 3.

Regarding the PSP multimodal optimization, it is known that methods with random or poor models' initialization generally tend to provide equally poor final results (JAIN; MURTY; FLYNN, 1999; GLIBOVETS; GULAYEVA, 2013). That is due to the inefficiency of the method in exploring the problem search space since random solutions present a higher tendency to get stuck in unfavorable regions throughout the process. Thus, methods with properly search space exploration, with the ability to generate and maintain distinct solutions during the process, are fundamental to overcome such scenarios. Moreover, the optimization process's initialization strategies capable of generating reasonable solutions are fundamental to support the search methods.

With this, the sampling initialization stage was idealized to initially explore the conformational search space considering some insights about previously known protein structures, aiming to locate different structural groups for the target protein. Besides, the solution initialization through the APL can be seen as a strategy to ease the optimization by restricting conformational possibilities. The sampling and classification of several individuals in clusters aim to increase the population diversity and exploration since the beginning of the method's execution.

In this step, 10,000 initial solutions are generated based on the input data of the target protein, using aa conformational preferences from the combination of the different APL types. We note that this number of sampling was defined according to our limitations of available memory hardware and computational power. The structures resulting from the sampling process are filtered according to the structural threshold known as the radius of gyration (RG) (LOBANOV; BOGATYREVA; GALZITSKAYA, 2008). The RG is used to infer the packaging structural state of the models. The RG threshold is established conforming specific characteristics of the target protein, which consider the aa sequence size and its structural class, established according to the SS arrangements. The maximum threshold is defined from the target protein size and class by analyzing experimentally determined protein structures that follow these same patterns (size and class). The RG of a protein structure is defined as the quadratic mean distance between all the protein atoms and its center of mass. It can be used as a packaging indicator since the lower the RG, the greater the proximity of atoms with the protein center of mass (LOBANOV; BOGATYREVA; GALZITSKAYA, 2008). From a biological point of view, if a protein structure is stable in its native state, the RG tends to remain stable. However, when the protein is out of its native state (less stable conformation), RG values tend to vary much more frequently. In this work, the RG calculation was performed through the PyRosetta libraries (CHAUDHURY; LYSKOV; GRAY, 2010).

The protein structures resulting from the filtering procedure are clustered according to their structural similarities through the Root-Mean-Square-Deviation (RMSD) measure (ZHANG; SKOLNICK, 2004). RMSD is used to assess the similarity degree between two protein structures. The formed groups are ranked considering the average group RG. The ones with the lowest values are used as initial individuals of the metaheuristic.

Initialization of Solutions

The sampling of solutions consists of initializing individuals for the metaheuristic, considering the conformational preferences of the amino acids of the target protein via the APL strategy. As previously described, the APL was divided into APL-neighborhood and APL-centroid, wherein in each of these divisions, there are subtypes of APLs that consider different combinations of amino acids to define the conformational preferences of the aa_{ref}. The APL-neighborhood takes into account the neighborhood of the aa_{ref} and their respective SS, forming combinations of 1, 2, or 3 amino acids. Contrarily, the APL-centroid only considers the SS of the amino acids present in the neighborhood of the aa_{ref}, without considering them, generating combinations of 5, 7, or 9 amino acids.

Therefore, each aa of the target protein is initialized through one of the two APLs (APL-neighborhood or APL-centroid), giving both the same chance of being chosen, i.e., probability of 50% for each one. It is noted that the larger the size of the APL combination, the more specific and restricted is the list of angular probabilities for a given aa_{ref}. For this reason, the APL subtypes are chosen according to the specificity order, that is, from more specific (larger combinations), with higher chances to be chosen, to less specific data (smaller combinations). Based on this, if the APL-neighborhood is chosen, the probability of 70% is given to APL-3, 20% to APL-2l and APL-2r, both of which with the same chance to be selected (50% for each), and 10% to APL-1. The same probability scheme is applied when APL-centroid is chosen, which means that the chance of 70% is given to APL-9, 20% to APL-7, and 10% to APL-5. We note that APL-1 and APL-5 encompass all data contained in the other more specific APLs. It is also observable that depending on the position of the aa_{ref}, not all types of APLs can be applied, which means that amino acids located at the extremes of the aa sequence can only be initialized through APL-1 or APL-2r, concerning the beginning of the sequence, or APL-1 or APL-2l, concerning the end of the sequence, since the other types of combinations do not fit into these positions.

The Algorithm 9 describes the procedure responsible for selecting which APLs will be employed to assign the torsion angles to a given aa, as described above. The function $QueryAPL$ (Algorithm 9, lines 4-24) represents the selected APL combination for the aa_{ref} based on the defined probabilities, target protein primary and secondary sequences (PS and SS), and the aa_{ref} position in the protein sequence. The function $rand(0, 1)$ generates a random value in the range $[0, 1]$. The algorithm outcomes a list of dihedral angles for the aa_{ref}.

This APL selection scheme was defined in the work of Corrêa and Dorn (CORRÊA; DORN, 2020) from a sample analysis of 100,000 individuals for a set of 8 target proteins. The authors generated the models by using: (*i*) only the APL-1; (*ii*) only the APL-neighborhood, considering the probabilities described above for its subtypes; (*iii*) only the APL-centroid, also considering the probabilities described above for its subtypes; and (*iv*) the APL-neighborhood and APL-centroid combined. The generated individuals were compared by the RMSD regarding the experimental structures and the RG measure. By the conducted analyses, the authors stated that no superiority was observed between the different APLs regarding the quality of the individuals created. However, the fact that the combination of APLs can provide a broader range of experimental data has motivated its use over others. Thus, by the sampling process of 10,000 solutions, this larger data set becomes more interesting since it provides a greater exploration of the experimental data characteristics, resulting in more diversified solutions. We note that the combination of APLs is referred to as APL-combined.

Filtering Process

The filtering of solutions was included after the initialization of solutions aiming to remove low-quality structures. This process intends to exclude such structures from the clustering process and also from the definition of structural groups used in the initialization of the metaheuristic. More generally, poor structures are characterized by the lack of packing, which tends to represent that they are distant from their native states. Thus, the resulting structures from the sampling process are filtered based on the RG threshold, a structural evaluation measure already mentioned. The RG is used, in this case, as an indicator of the packing level of structural models. The threshold value was defined in the work of Corrêa and Dorn (CORRÊA; DORN, 2020) by the analysis of 25,135 proteins extracted from the PDB. This set of proteins has been devised following the same filtering elements used in the assembly of the APL database (BORGUESAN; INOSTROZA-PONTA; DORN, 2016; CORRÊA; DORN, 2020). These filters are summarized in Table 5.2.

Table 5.2: Set of filters applied in the generation of the database used to define the RG threshold

Filter	**Thresholds**
Protein set length	25,135 structures
Resolution	≤ 2.5
R-observed	≤ 0.20
Similarity of sequences	$\leq 30\%$
B-factor	$\leq 30^2$
Occupancy	1

Source: Adapted from Corrêa and Dorn (CORRÊA; DORN, 2020).

Specifically, regarding the creation of the database mentioned above, all the polypeptide chains or subunits (quaternary structure) included in the PDB files were considered. On the contrary, in the structuring of the APL database, only the first polypeptide chain of each structure in the PDB file was maintained (BORGUESAN; INOSTROZA-PONTA; DORN, 2016; CORRÊA; DORN, 2020). For the database used in this step, the maintenance of equal chains does not influence the definition of the RG thresholds, and for this reason, all chains were considered. For each protein in the database, its SS was assigned by the STRIDE software, and the values of RG were computed. The database relies on 25,135 protein structures of several sizes, ranging from 5 to 3,680 amino acids. The variation of RG values comprises the range of 5.58 to 72.03.

As described in Chapter 3, proteins can be classified according to their SS arrays. To obtain more specific information about the proteins considered in the idealized protein database, the authors classified the proteins according to their SS, resulting in the scenario described in Table 5.3. Proteins were classified into four distinct classes, which comprise:

1. Class of less stable regions, which encompass structures with more than 80% turns or coils in their SS arrays;
2. Class of β-sheets, comprising proteins which have a predominance of more than 60% of β-sheets in their SS;
3. Class of helices, encompassing protein structures with more than 60% of helices in their SS;
4. Hybrid class, which comprises structures that do not fit into any of the above classes, i.e., structures that present a combination of the three SS types in their arrays.

According to Table 5.3, we note that the hybrid class holds the most significant number of protein structures.

Table 5.3: Summary of the different classes of proteins generated from the PDB database

Information data	
Class of less stable regions	
Definition	Predominance of 80% of turns or coils
Set Size	1,375 structures
Size variation	[5 aa, 1,656 aa]
RG variation	[5.58, 72.03]
Class of β-sheets	
Definition	Predominance of 60% of β-sheets
Set Size	324 structures
Size variation	[12 aa, 598 aa]
RG variation	[7.6, 39.8]
Class of helices	
Definition	Predominance of 60% of helices
Set Size	3,797 structures
Size variation	[6 aa, 1,184 aa]
RG variation	[6.15, 65.51]
Hybrid class	
Definition	Hybrid structures
Set Size	19,639 structures
Size variation	[5 aa, 3,680 aa]
RG variation	[5.89, 62.59]

Source: Adapted from Corrêa and Dorn (CORRÊA; DORN, 2020).

Regarding the correlation between the size of experimentally determined proteins and their RG values, theoretically, the higher the number of amino acids, the higher the RG values since the 3-D protein structure is bigger. However, according to the work of Corrêa and Dorn (CORRÊA; DORN, 2020), some proteins do not follow this pattern. It can be attributed to the different arrangements of SS and conformations in the database and the specific physicochemical properties of the amino acids that constitute the proteins. With this, it is important to consider such components in filtering solutions. The classification of proteins into classes and the correlation between the size of amino acid sequences and their RG values can improve the discovery process of specific characteristics of the target protein based on previously known protein structures.

Thus, for a given target protein, the maximum RG threshold is defined by querying the database, correlating the size of the amino acid sequence and its class. From the set of experimental proteins returned by this query, the threshold is defined from the highest RG value linked to the returned structures. We note that the proteins returned by the database query have the same size and class as the target protein. Finally, after defining the RG threshold, the structures resulting from the sampling process described in the previous section are filtered based on this packing indicator. Structures that exceed the defined threshold are removed from the process.

To analyze the behavior of the filtering procedure, Corrêa and Dorn (CORRÊA; DORN, 2020) sampled 10,000 individuals considering a small set of 8 target proteins. The sampling processes were conducted by applying and not applying the filtering procedure. Generated individuals were compared through RMSD regarding the experimentally determined structures. From this analysis, the authors stated that there is a significant reduction in the number of solutions for some targets due to the filtering and discard of solutions. In contrast, the majority are poor solutions with high RMSD. So, the procedure provided a smaller and better set of solutions than the sampling process that does not use the exclusion thresholds. The sets of solutions from the filtering process that used the RG threshold tend to encompass structures with relatively lower RMSD values when compared with the excluded solutions. But we note that these structures are also in the sets of solutions that were generated without considering the threshold. However, these sets still encompass the highest RMSD solutions that have been discarded from the sets with RG constraints. For this reason, the filtering process can be considered a simple but efficient way to remove low-quality structures before the optimization process through the metaheuristic.

Clustering of Solutions

As the last step of the sampling and initialization stage, the clustering process of structural models aims to identify the different conformational clusters generated from the sampling process. Data clustering is an important technique based on unsupervised learning, which is defined as the process of clustering objects taking into account their patterns of similarity and dissimilarity. It aims to cluster, in the same group, elements with a high degree of similarity and, in different groups, elements with a low degree of similarity (JAIN; MURTY; FLYNN, 1999). This procedure supports the discovery of specific characteristics related to the target protein, e.g., the clustering of individuals throughout the optimization process can reveal conformational patterns, local minimum basins and indicate the algorithm's convergence. This strategy is commonly adopted among reference methods in the field, as shown in Rosetta (ROHL et al., 2004).

Thus, the resulting structures from the filtering process are clustered according to their structural similarities using the RMSD, aiming to identify distinct conformational patterns for the target protein. The clusters formed are ranked considering the average group RG, and those that present the lowest values are used as initial individuals of the metaheuristic, highlighting the structural packing. We observe that the clusters could also be ranked according to the number of individuals in each group or the average energy values. However, according to a clustering analysis performed using these metrics in Corrêa and Dorn (CORRÊA; DORN, 2020), the average RG was able to order the formed clusters so that the best solutions tended to be in the first groups, which give rise to the initial population of the metaheuristic. So, in this work, we adopted the average RG of the clusters to rank the most promising structural groups.

The clustering was performed through the agglomerative hierarchical clustering technique (JAIN; MURTY; FLYNN, 1999). Hierarchical data clustering generates a sequence of nested groups, forming a hierarchy structure between them. The highest level of the hierarchy consists of a single group that comprises all other partitions formed, and the lowest level includes each object individually allocated in a single group (JAIN; MURTY; FLYNN, 1999). Agglomerative clustering is a bottom-up process where each object in the data set initially represents a single group. The most similar groups are clustered by some similarity measure at each clustering step. At the end of the process, all formed clusters represent a single group. The distance measure used to evaluate the degree of similarity between protein models was the RMSD. The criterion for evaluating the distance between clusters (inter-cluster distance) was calculated by the complete linkage clustering, which considers the largest distance between two objects of different groups. The cut-off threshold adopted in the clustering imposes the formation of at least N_{nodes} clusters since each one must have Max_NS/N_{nodes} or more structures (individuals). Max_NS and N_{nodes} represent the maximum number of individuals allowed in the entire population of the MA and the number of nodes in the tree structure, respectively. These thresholds are related to the parameters of the MA framework, i.e., the population size and the tree structure. We note that the clustering was performed through the libraries provided by the SciPy[3] software.

Finally, the sampling and classification stage, which culminates in the clustering of the generated models in different structural clusters, is used to feed the MA subpopulations with better and diversified solutions to support the search algorithm in the next stage. Thus, the first stage intends to overcome the multimodality issues of the objective function by enhancing the quality of the initial individuals.

[3]<www.scipy.org>

5.4.3 Optimization of the Protein Structures

In this step, the previously described MA framework (Section 5.2) was incremented to deal with the PSP problem. This MA version was designed based on the algorithm presented in the work of Corrêa and Dorn (CORRÊA; DORN, 2020), which incorporated specific-problem operators such as recombination operations considering the target protein SS and constraints in the mutation of individuals based on conformational preferences of amino acids. Each node of the tree structure is initialized with a structural cluster from the previous step.

Regarding the PSP, the designed MA encompasses niching strategies, interactions between distinct populations, intra-node optimization, and control procedures for local improvements, convergence, and performance. The algorithm aims to overcome the PSP multimodal energy function issues by regulating the Expr/Expt mechanisms and the solution improvements throughout the process. The subpopulations of the MA are optimized independently by the hybrid-ABC (Section 5.2.12) and the framework as a whole by the interactions defined between the nodes of the tree. Our goal is to find and maintain distinct and reasonable solutions located in different regions of the search space.

As already mentioned, the MA was implemented to favor both exploration and search space refinement according to the stage of the evolutionary process, such as population diversity degree and solution improvements. However, some components of the algorithm were adapted to tackle the problem, such as the crossover operator to enhance its accuracy by using the knowledge from known protein structures and the constraints incorporated in the mutation operations to avoid unfavorable regions of the conformational search space, indicated by the conformational preferences of amino acids from APL-1. These modifications in the algorithm are detailed in the following sections.

Algorithmic Structure of the Method for the PSP Problem

The general structure of the MA framework applied to the PSP problem was previously described in Section 5.2. The MA was structured as a ternary tree data structure with N_{nodes} nodes. It consists of a multi-population technique, which arranges the population of individuals in independent subpopulations into the tree's nodes. The algorithmic structure of the framework was detailed in Section 5.2.1, and its optimization steps are shown in Section 5.2.3. Figure 5.1 shows the MA general flowchart.

Nonetheless, some modifications in its components were done to deal with the problem. The subpopulations of the tree are initialized by the structural groups generated in the first stage of sampling and initialization of solutions. After the initialization of the method, the steps described in Section 5.2.3 are executed in each generation. The stop criterion is determined by the maximum number of energy calculations (i.e., fitness function evaluations) performed throughout the optimization. The MA version for the PSP receives the target protein primary and secondary structures and the maximum number of energy calculations to be performed as input parameters. It returns the best solution for each subpopulation. It is noted that the parameter setting of this algorithm is described in the section of computational experiments (Section 6.4).

Representation of Individuals

As described in Section 3.6.2 concerning the computational modeling of the PSP, each aa of the target protein can be represented by a vector of seven real values. Three of them represent the dihedral angles ϕ, ψ, and ω of the main chain, and the remaining values represent the dihedral angles χ of the side chain.

Nonetheless, as we are interested in the overall folding of protein models, in this work, we adopted the centroid-based representation model for energy evaluations (ROHL et al., 2004), described in Section 3.5. In this representation, the main chain remains fully atomic. However, the representation of each aa side chain is simplified to a single pseudo-atom arranged in the side chain center of mass. The centroid representation simplifies the side chain complexity while keeping the overall protein folding by preserving the backbone integrity. Thus, the centroid-based objective function of Rosetta was employed, and the model conversion between the dihedral angle representation to the atomic coordinate is done by the own Rosetta's energy function implementation (ROHL et al., 2004; CHAUDHURY; LYSKOV; GRAY, 2010). With this, the computational representation of a given solution X with n amino acids is defined by a vector of real values of size $n \times 3$, according to the Equation 5.17.

$$X = [x_{1_\phi}, x_{1_\psi}, x_{1_\omega}, \cdots, x_{n_\phi}, x_{n_\psi}, x_{n_\omega}] \tag{5.17}$$

Recombination Operator

Recombination operations are performed globally between distinct subpopulations of the MA (Section 5.2.5) and internally in each node of the tree (Section 5.2.4), as previously described. To be more effective in exploring intrinsic problem-properties, the crossover operator proposed by Corrêa et al. (CORRÊA et al., 2016) was adopted instead of the Uniform crossover.

The Uniform SS crossover aims to maintain the similarity found during the optimization process between the SS of the solutions being optimized and the SS previously informed as an input parameter of the algorithm. This strategy aims to replicate the correct SS arrangements of parents in children.

Firstly, two solutions are selected as parents for the recombination. We observe that the selection and replacement strategies are the same as the general MA. Similar to the Uniform crossover, for every aa of the target protein (set of angles x_i in the vector X of Equation 5.17), the angles are chosen from both parent 1 or parent 2 (with 50% probability). This procedure occurs if both SS regarding the parents are the same or different from the SS informed as input parameter for the respective aa. If only one of the parents has SS equal to the input SS for a specific aa, then the set of angles corresponding to the concerned aa belonging to this parent is chosen. All target protein amino acids are compared during the operation.

Mutation Restriction

To better adapt the method to the problem characteristics, we adapted the ABC mutation operator to generate feasible candidate solutions and guide the search towards promising regions of the search space. It consists of verifying the new values generated by the update operation of the algorithm. At each mutation of the objective variable v_{ij}, it is checked whether the newly generated value is in APL-1, which indicates the conformational preferences of the related aa. This procedure aims to restrict unfavorable angle values or even out of the continuous range $[-180, 180]$. Thus, if the new assumed value is not in APL-1, it is not considered, and the old value remains.

5.5 Final Remarks

This chapter described the algorithms and search strategies used in this thesis to deal with optimization problems regarding the multimodal continuous domain, as well as the methodology of the proposed methods. With the development of the methods via an incremental approach, our focus was to create a robust method and evaluate its behavior facing different multimodal optimization functions and scenarios.

Therefore, the first designed method was proposed for continuous optimization but focused on global optimization with multimodal objective functions. It was implemented as a general optimization framework that is easily adapted to other search strategies and components. The method was structured based on a hierarchical tree data structure implemented under an adaptive framework-based MA. It consists of a multi-population technique, which arranges the population of individuals in subpopulations into the tree's nodes through a clustering-based niching strategy. The proposed MA encompasses independent intra-node optimization through a core search metaheuristic and interactions between distinct populations as a way of knowledge sharing, population diversification, and exploration inter and intra-niches. As the core metaheuristic, a hybrid-ABC algorithm was presented, where each node of the tree incorporates an independent execution.

The framework also includes control procedures of adaptive and hybrid approaches for local improvements, population convergence and performance, and dynamic parameter adaptation. It aims to overcome the multimodality issues of the search space by considering the status of the optimization process during the algorithm's execution and adjusting the search mechanisms to it, that is, the population diversity and convergence degree, Expr/Expt efforts, and the performance of the algorithm by improving solutions. Then, this MA framework was used as a basis for developing other designed versions to deal with multimodal optimization and tackle the PSP problem.

The second variant of the MA is focused on multimodal optimization when dealing with objective functions with more than one global optimum. The main issues of multimodal optimization are related to discovering and maintaining the existing global optima of a given problem while locating new ones throughout the search process. Thus, the first version of the method was modified based on such concerns. This version incorporates a simple external archive to store the identified global optima when converged regions of the search space are detected. Utilizing the MA strategies for control converged individuals, whenever a discard of solutions occurs, the individuals are stored into the archive to keep the found promising solutions separate from the current population. In contrast, the framework generates new solutions to continue exploring new areas.

Finally, the last version of the method was developed to deal with the PSP problem. The framework for the PSP has enhanced with the problem-dependencies and data knowledge from experimentally determined 3-D protein structures to turn it more robust for the problem. The method can be divided into two main optimization stages. The first step of sampling and initializing structural models is responsible for generating and classifying several models from the aa primary sequence of a target protein. The models are created from the APL strategy, considering the aa probability of occurrence based on conformational preferences of previously known protein structures. Also, the sample of individuals is submitted to filtering and clustering processes, aiming to highlight distinct structural patterns and create a better and diversified sample of protein models. It intends to reduce the search space size, overcome its roughness and provide feasible initial solutions to the second optimization step.

The second step represents the optimization of solutions sampled from the first stage. The MA framework performs the search process, which was incremented to deal with the problem. The method includes modified specific-problem operators, such as the Uniform SS crossover and the constraints in the mutation operations based on conformational preferences of amino acids. Such strategies aim to increase the method's performance and the accuracy of the solutions, focusing on the characteristics of the PSP.

Lastly, all of the proposed methods were implemented following the points of interest outlined for this thesis, which include search space exploration to discover and maintain the existing optimal solutions, a reasonable balance between Expr/Expt by controlling the population diversity, and parameter setting control to deal with global and multimodal optimization functions. It is known that each optimization function requires distinct abilities from algorithms, even in the same application domain. Then, we expected to enhance our approach with such a need without degrading previously incorporated search strategies. The next chapter presents the obtained results and analyses from the optimization problems through the developed algorithms presented in this chapter. Moreover, the incremental design, the MA versions, and parameterization designed to reach the constructive versions presented in this chapter are also discussed.

Algorithm 7 Pseudocode of the framework-based MA.

```
Require: $N_{layers}$: number of layers; $Max\_NS$: maximum population size; $Max\_Evls$: maximum number of fitness evaluations; $g_{core}$: number of core search generations; $n\_gen$: period of generations to calculate the SMAs
Ensure: $nodes$: optimized node subpopulations
 1: $N_{nodes} \leftarrow \sum_{i=0}^{N_{layers}-1} 3^i$; $p_{best}\% = 25$; $n_{frec} \leftarrow 1000$; $rm_{layer} = Max\_Evls/N_{layers}$; $evol_{time} = 1$; $gen_{curr} \leftarrow 0$; $f\_evls \leftarrow 0$; $not_imp \leftarrow 0$
 2: $pop \leftarrow$ randomly initialize the $Max\_NS$ solutions
 3: $nodes \leftarrow ClusteringSpeciation(pop, Max\_NS, N_{nodes})$ //Niching strategy
 4: calculate $Div\_initial_{combined}$ for each $node_i$ in $nodes$ (Equation 5.4)
 5: while $f\_evls < Max\_Evls$ do
 6:     for each $node_i$ in $nodes$, $i \leftarrow 1 : N_{nodes}$ do
 7:         for $1 : g_{core}$ do
 8:             apply $CoreMetaheuristic$ in $node_i$
 9:             apply $InnerRecombination$ in $node_i$
10:         end for
11:         if $nodes_i$ is a parent then //Interactions between nodes
12:             apply bottom-up crossover using $node_i$ as parent (Algorithm 3)
13:             apply top-down crossover using $node_i$ as parent (Algorithm 3)
14:         end if
15:         if $n_{frec} \times N_{nodes}$ fitness function evaluations were performed then //LIP
16:             apply $LSchain$ in one of the $p_{best}\%$ best solutions of $node_i$
17:             if any $p_{best}\%$ best solutions of $node_i$ was improved then
18:                 apply $LS\_restarting$ in $node_i$ //LS restarting considering the priority order
19:             end if
20:             restart the count for fitness function evaluations
21:         end if
22:         if $n\_gen$ generations were performed then //Restarting considering the priority order
23:             $VerifyRestarting(nodes, Div\_initial_{combined}, n\_gen)$ (Algorithm 4)
24:         end if
25:     end for
26:     if $rm_{layer}$ fitness function evaluations were performed then //TRP
27:         $N_{layers} \leftarrow N_{layers} - 1$; $N_{nodes} \leftarrow \sum_{i=0}^{N_{layers}-1} 3^i$ //TRP considering the priority order
28:         restart the count for $rm_{layer}$
29:     end if
        //Control mechanism for convergence and performance considering the priority order
30:     determine the $gbest$ solution
31:     if $gbest$ was improved then
32:         $not\_imp = 0$
33:     else
34:         $not\_imp = not\_imp + 1$
35:     end if
36:     apply $PerformanceVerification(not\_imp, evol_{time}, evol_{previous}, gen_{curr})$ (Algorithm 5)
37:     if $n\_gen$ generations were performed then //DNSP
38:         apply $DynamicNicheSize(nodes, Expr\%, n\_gen)$
39:         restart the count for $n\_gen$
40:     end if
41:     Update $gen_{curr}$ and $f\_evls$
42: end while
43: return $nodes$
```

Algorithm 8 Pseudocode of the proposed hybrid ABC algorithm.

```
Require: $pop$: subpopulation to be optimized; $g_{core}$: stop criterion
Ensure: $pop$: optimized subpopulation
 1: $MR \leftarrow 0.4$; $l \leftarrow SN \times D$
 2: for $gen \leftarrow 1 : g_{core}$ do
 3:     $md = (f(pop_{worst}) + f(pop_{best}))/2$ //Population division threshold
        //Employed bees' stage
 4:     for each $pop_i$ in $pop$, $i \leftarrow 1 : SN$ do
 5:         if $f(pop_i) > md$ then //Superior level
 6:             define the values for $\delta_{i,gen+1}$ and $\gamma_{i,gen+1}$ by using Equations 5.12 and 5.13
 7:             for each $pop_{ij}$, $j \leftarrow 1 : D$ do
 8:                 if $(rand(0,1) < MR)$ then
 9:                     generate a new value for $v_{ij}$ from $pop_{ij}$ by $\delta_{i,gen+1}$, $\gamma_{i,gen+1}$, Equation 4.2
10:                 end if
11:             end for
12:             if $v_i$ is better than the original $pop_i$ then
13:                 replace $pop_i$ for $v_i$; update $\delta_{i,gen}, \gamma_{i,gen} \leftarrow \delta_{i,gen+1}, \gamma_{i,gen+1}$
14:             end if
15:         else //Inferior level
16:             define the value for $F_{i,gen+1}$ by using Equation 5.14
17:             for each $pop_{ij}$, $j \leftarrow 1 : D$ do
18:                 if $(rand(0,1) < MR)$ then
19:                     generate a new value for $v_{ij}$ from $pop_{ij}$ by $F_{i,gen+1}$ and Equation 4.4
20:                 end if
21:             end for
22:             if $v_i$ is better than the original $pop_i$ then
23:                 replace $pop_i$ for $v_i$; update $F_{i,gen} \leftarrow F_{i,gen+1}$
24:             end if
25:         end if
26:     end for
27:     $md = (f(pop_{worst}) + f(pop_{best}))/2$ //Population division threshold
        //Onlooker bees' stage
28:     for $i \leftarrow 1 : SN$ do
29:         $indv \leftarrow$ ranking-based selection of solution from $pop$
30:         if $f(indv) > md$ then //Superior level
31:             apply to $indv$ the same update operation for superiors of the previous stage
32:         else //Inferior level
33:             apply to $indv$ the same update operation for inferiors of the previous stage
34:         end if
35:     end for
        //Scout bees' stage
36:     for each $pop_i$, $i \leftarrow 1 : SN$ do
37:         check if solution $pop_i$ has not been improved over $l$ generations
38:         discard $pop_i$
39:         insert a new solution in the population
40:     end for
41:     apply $InnerRecombination$ in $pop$ from the MA framework
42: end for
43: return $pop$
```

Algorithm 9 Definition of APL to be used in the assignment of torsion angles to a given amino acid.

Require: PS, SS: primary and secondary amino acid sequences; $pos_{aa_{ref}}$: position of aa_{ref}
Ensure: $angles$: set of torsion angles for the aa_{ref}

```
    //Selection between APL-neighborhood and APL-centroid
 1: if $0.5 \leq rand(0,1)$ then
      //APL-neighborhood
 2:   $prob \leftarrow rand(0,1)$
 3:   if $prob \leq 0.7$ then
 4:     $angles \leftarrow QueryAPL3(PS, SS, pos_{aa_{ref}})$
 5:   else
 6:     if $prob \leq 0.9$ then
 7:       if $0.5 \leq rand(0,1)$ then
 8:         $angles \leftarrow QueryAPL2l(PS, SS, pos_{aa_{ref}})$
 9:       else
10:         $angles \leftarrow QueryAPL2r(PS, SS, pos_{aa_{ref}})$
11:       end if
12:     else
13:       $angles \leftarrow QueryAPL1(PS, SS, pos_{aa_{ref}})$
14:     end if
15:   end if
16: else
      //APL-centroid
17:   $prob \leftarrow rand(0,1)$
18:   if $prob \leq 0.7$ then
19:     $angles \leftarrow QueryAPL9(PS, SS, pos_{aa_{ref}})$
20:   else
21:     if $prob \leq 0.9$ then
22:       $angles \leftarrow QueryAPL7(PS, SS, pos_{aa_{ref}})$
23:     else
24:       $angles \leftarrow QueryAPL5(PS, SS, pos_{aa_{ref}})$
25:     end if
26:   end if
27: end if
28: return $angles$
```

6 COMPUTATIONAL EXPERIMENTS

6.1 Introduction

We designed three versions of the MA framework via a constructive approach for three different scenarios of multimodal optimization: (*i*) the general MA framework for single global continuous optimization with multimodal fitness function; (*ii*) the MA framework with archive strategy for multimodal optimization with more than one global optimum; and (*iii*) the MA framework with specific-problem components for the multimodal problem of predicting the 3-D protein structures. All the algorithms described in this thesis were coded in Python. We observe that the stop criterion (maximum number of fitness function evaluations) for each problem depends on its respective optimization scenario, which is detailed in the following sections. Tests were performed in an Intel Xeon E5-2650V4 30 MB, 4 CPUs, 2.2Ghz, 96 cores/threads, 128G, 4TB.

The chapter describes the benchmark test functions employed in each scenario, the algorithms used for comparison, the parameter setting, and the metrics applied for evaluation. Lastly, we present the obtained results regarding the proposed MA framework's performance facing each optimization case study.

6.2 Scenario of Single Global Optimization

As discussed in Section 4.5, global continuous optimization algorithms represent the foundation for more complex scenarios, such as niching algorithms in multimodal optimization (SER et al., 2019). According to Price et al. (PRICE et al., 2018), single objective benchmark problems are the first tests for novel metaheuristics since such problems can be transformed into dynamic, niching composition, computationally expensive, and other classes of problems.

The general MA framework was tested on the 100-Digit Challenge on single objective real-parameter optimization to test the behavior of it as a single objective optimizer[1] (PRICE et al., 2018) adopted in the CEC 2019, GECCO 2019 and SEMCCO 2019 (PRICE et al., 2019). This benchmark encompasses ten minimization problems, which consist in ten functions to minimize, such that $\min f(x), x \in \mathbb{R}^D$, where x is a D-dimensional vector $[x_1, \cdots, x_D]$. The benchmark suite is detailed in Table 6.1. It

[1]<https://github.com/P-N-Suganthan/CEC2019>

is noted that we used the Python implementation of the test suite available on <https://github.com/dmolina/cec2019comp100digit>.

Table 6.1: Summarization of the 100-Digit Challenge benchmark test functions

Function ID	**Function**	$F_i* = F_i(x*)$	***D***	**Search range**
`F1`	Storn's Chebyshev Polynomial Fitting Problem	1	9	[-8192, 8192]
`F2`	Inverse Hilbert Matrix Problem	1	16	[-16384, 16384]
`F3`	Lennard-Jones Minimum Energy Cluster	1	18	[-4,4]
`F4`	Rastrigin's Function	1	10	[-100,100]
`F5`	Griewank's Function	1	10	[-100,100]
`F6`	Weierstrass Function	1	10	[-100,100]
`F7`	Modified Schwefel's Function	1	10	[-100,100]
`F8`	Expanded Schaffer's F6 Function	1	10	[-100,100]
`F9`	Happy Cat Function	1	10	[-100,100]
`F10`	Ackley Function	1	10	[-100,100]

Source: From Price et al. (PRICE et al., 2018).

According to Table 6.1, all functions are multimodal with dimensions (objective variables) varying from 9-D to 18-D. The challenge's goal is to compute each function's minimum value to 10 digits of accuracy without being limited by time or maximum number of function evaluations. The optimal objective variables are known for all benchmark functions, and the global optimum for all functions to 10 digits of accuracy is 1.000000000. The last column shows each function's upper and lower bounds, where the same search range is defined for all function dimensions. The main characteristics of the benchmark problems are given as follows (PRICE et al., 2018):

F1: Storn's Chebyshev Polynomial Fitting Problem (Equation 6.1):

$$\begin{gathered}
F_1(x) = p_1 + p_2 + p_3; \\
p_1 = \begin{cases} (u-d)^2 & \text{if } (u < d), \\ 0 & \text{otherwise;} \end{cases} \quad u = \sum_{j=1}^{D} x_j (1.2)^{D-j} \\
p_2 = \begin{cases} (v-d)^2 & \text{if } (v < d), \\ 0 & \text{otherwise;} \end{cases} \quad v = \sum_{j=1}^{D} x_j (-1.2)^{D-j} \\
p_k = \begin{cases} (w_k-1)^2 & \text{if } (w_k > 1), \\ (w_k+1)^2 & \text{if } (w_k < 1), \\ 0 & \text{otherwise;} \end{cases} \quad w_k = \sum_{j=1}^{D} x_j \left(\frac{2k}{m} - 1\right)^{D-j} \\
p_3 = \sum_{k=0}^{m} p_k, \quad k = 0, \cdots, m; \quad m = 32D; \\
d = 72.661 \text{ for } D = 9
\end{gathered} \tag{6.1}$$

Function properties:

- Multimodal with one global minimum;
- Very highly conditioned;
- Non-separable and fully parameter-dependent.

F2: Inverse Hilbert Matrix Problem (Equation 6.2):

$$
\begin{gathered}
F_2(x) = \sum_{i=1}^{n}\sum_{k=1}^{n} \mid w_{i,k} \mid \\
(w_{i,k}) = \mathbf{W} = \mathbf{HZ} - \mathbb{I}; \\
H(h_{i,k}), \quad h_{i,k} = \frac{1}{i+k-1}, \quad i,k = 1,2,\cdots,n; \quad n = \sqrt{D}; \\
Z(z_{i,k}), \quad z_{i,k} = x_{i+n(k-1)}
\end{gathered}
\tag{6.2}
$$

Function properties:

- Multimodal with one global minimum;
- Highly conditioned;
- Non-separable and fully parameter-dependent.

F3: Lennard-Jones Minimum Energy Cluster (Equation 6.3):

$$
\begin{gathered}
F_3(x) = 12.7120622568 + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(\frac{1}{d_{i,j}^2} - \frac{2}{d_{i,j}}\right); \\
d_{i,j} = \left(\sum_{k=0}^{2}(x_{3i+k-2} - x_{3j+k-2})^2\right)^3, \quad n = D/3
\end{gathered}
\tag{6.3}
$$

Function properties:

- Multimodal with one global minimum;
- Non-separable and fully parameter-dependent.

F4: Shifted and Rotated Rastrigin's Function (Equation 6.4):

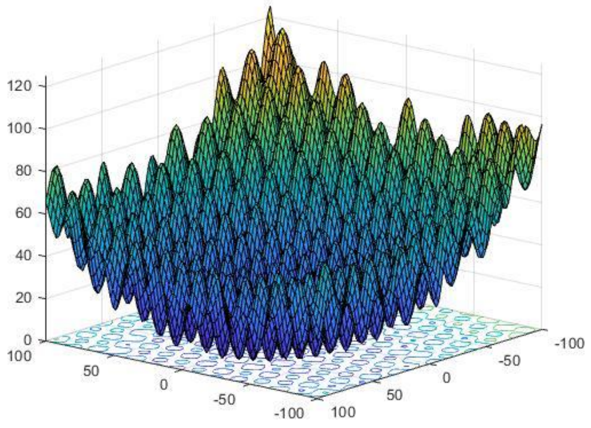
$$
F_4(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10) \tag{6.4}
$$

Function properties:

- Multimodal;

- Non-separable;
- Huge number of local optima;

Figure 6.1: 3-D landscape representation for 2-D F4



Source: From Price et al. (PRICE et al., 2018).

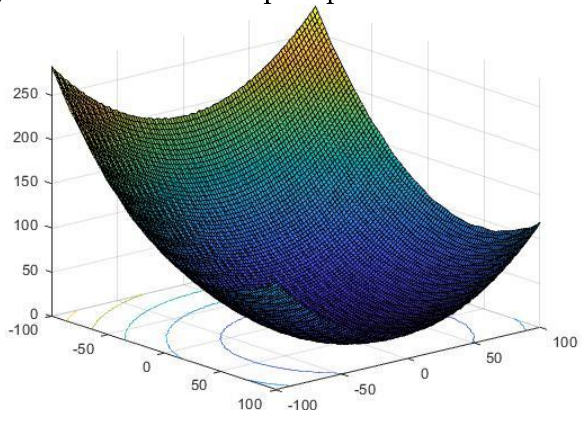
F5: Shifted and Rotated Griewank's Function (Equation 6.5):

$$F_5(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{6.5}$$

Function properties:

- Multimodal;
- Non-separable;

Figure 6.2: 3-D landscape representation for 2-D F5



Source: From Price et al. (PRICE et al., 2018).

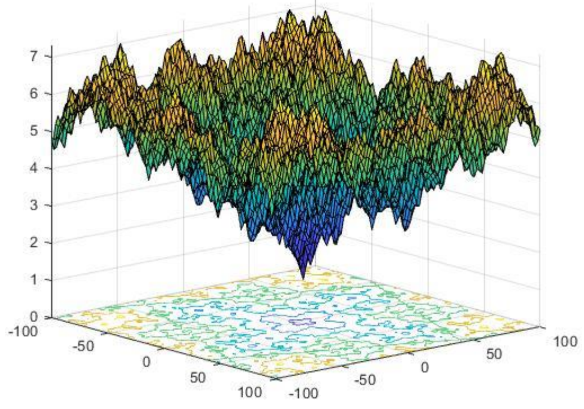
F6: Shifted and Rotated Weierstrass Function (Equation 6.6):

$$F_6(x) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} \left[a^k \cos\left(2\pi b^k (x_i + 0.5)\right) \right] \right) - D \sum_{k=0}^{k_{max}} a^k \cos(\pi b^k); \tag{6.6}$$
$$a = 0.5; \quad b = 3; \quad k_{max} = 20;$$

Function properties:

- Multimodal;
- Non-separable;
- Huge number of local optima;

Figure 6.3: 3-D landscape representation for 2-D F6



Source: From Price et al. (PRICE et al., 2018).

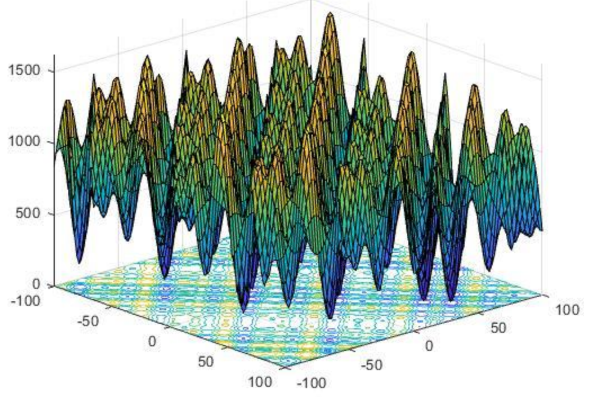
F7: Shifted and Rotated Schwefel's Function (Equation 6.7):

$$F_7(x) = 418.9829D - \sum_{i=1}^{D} g(z_i), \quad z_i = x_i + 420.9687462275036;$$
$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{1/2}) & \text{if } (|z_i| \leq 500), \\ (500 - z_i\%500) \sin\left(\sqrt{|500 - z_i\%500|}\right) - \frac{(z_i - 500)^2}{10000D} & \text{if } (z_i > 500), \\ (|z_i|\%500 - 500) \sin\left(\sqrt{|z_i\%500 - 500|}\right) - \frac{(z_i + 500)^2}{10000D} & \text{if } (z_i < -500); \end{cases} \tag{6.7}$$

Function properties:

- Multimodal;
- Non-separable;
- Huge number of local optima;

Figure 6.4: 3-D landscape representation for 2-D F7



Source: From Price et al. (PRICE et al., 2018).

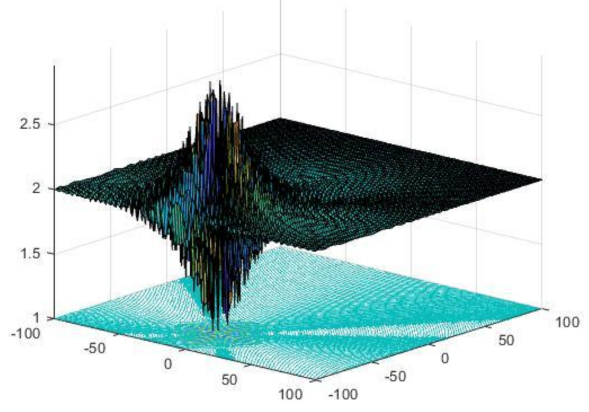
F8: Shifted and Rotated Expanded Schaffer's F6 Function (Equation 6.8):

$$F_8(x) = g(x_1, x_2) + g(x_2, x_3) + \cdots + g(x_{D-1}, x_D) + g(x_D, x_1);$$
$$g(x, y) = 0.5 + \frac{\sin^2\left(\sqrt{x^2 + y^2}\right) - 0.5}{(1 + 0.001(x^2 + y^2))^2} \tag{6.8}$$

Function properties:

- Multimodal;
- Non-separable;
- Huge number of local optima;

Figure 6.5: 3-D landscape representation for 2-D F8



Source: From Price et al. (PRICE et al., 2018).

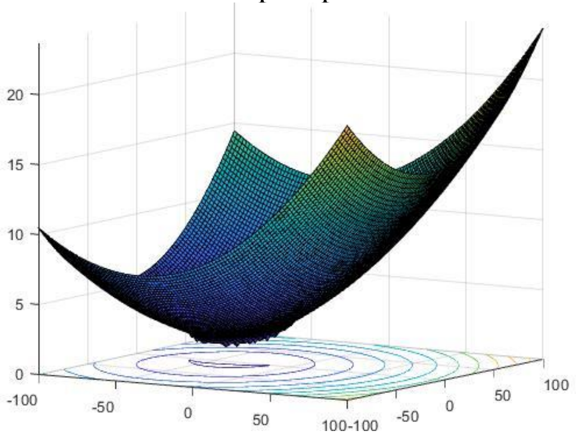
F9: Shifted and Rotated Happy Cat Function (Equation 6.9):

$$F_9(x) = \left|\sum_{i=1}^{D} x_i^2 - D\right|^{1/4} + \left(0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i\right) \Big/ D + 0.5 \tag{6.9}$$

Function properties:

- Multimodal;
- Non-separable;

Figure 6.6: 3-D landscape representation for 2-D F9



Source: From Price et al. (PRICE et al., 2018).

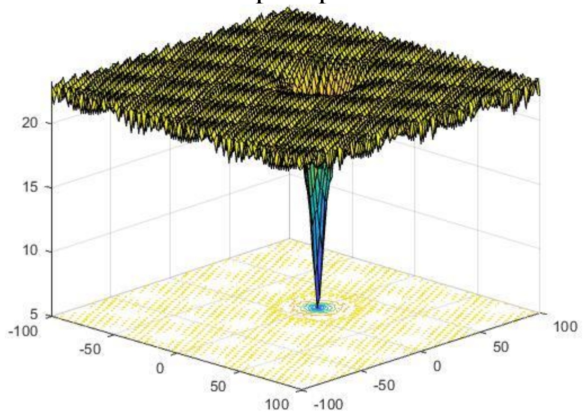
F10: Shifted and Rotated Ackley Function (Equation 6.10):

$$F_{10}(x) = -20\exp\left(0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e \tag{6.10}$$

Function properties:

- Multimodal;
- Non-separable;

Figure 6.7: 3-D landscape representation for 2-D F10



Source: From Price et al. (PRICE et al., 2018).

6.2.1 Evaluation Criteria

The benchmark functions shown in Table 6.1 represent ten minimization problems, where the goal is to compute each function's minimum value to 10 digits of accuracy using only one algorithm. As mentioned, the optimal objective variables are known for all functions, and the global minimum for all functions to 10 digits of accuracy is 1.000000000. All problems have the global optimum within the given bounds. According to the challenge's evaluation criteria (PRICE et al., 2018), there is no limit on either time or maximum number of fitness function evaluations. Thus, the stop criterion is determined when the algorithm reaches the 10-digit level of accuracy or the maximum number of function evaluations defined by the user.

Therefore, 50 consecutive runs of an algorithm are required for each function, each with a different initial population. The population is randomly initialized within the search space (search range column in Table 6.1). The total number of correct digits in the 25 runs with the lowest function values (the 25 runs with the best fitness values out of the 50 runs) is counted in the function's score. The score for that function is the average number of correct digits in the best 25 runs, i.e., if 50% or more of the runs find all 10 digits, then the score for that function is a perfect 10. The total score of an algorithm for the ten functions is the sum of scores of each function. Then, the maximum total score for the ten functions is 100, i.e., when the best 25 out of 50 runs for the ten functions give the minimum to 10-digit accuracy. The score for each function is given by Equation 6.11 and represents the average number of correct digits obtained in the best 25 runs of the function.

$$F_{score}(N_{Dcorr}) = \left(\frac{\sum_{i=1}^{25} N_{Dcorr_i}}{25} \right) \tag{6.11}$$

Where N_{Dcorr} is a vector with the number of correct digits obtained in each of the best 25 runs of a given function.

The total score of the method for all functions is given by Equation 6.12, which represents the sum of scores of the ten functions calculated individually by Equation 6.11.

$$F_{total\_score} = \sum_{i=1}^{10} F_{score_i} \tag{6.12}$$

Moreover, according to the benchmark specifications, the user may independently tune up to two parameters for each problem. Adaptive parameters do not count as tuned parameters, provided that they are both initialized and adapted identically for all ten problems. However, it is noted that we kept the parameter setting of our method the same for all functions since the method is used as the basis for the subsequent optimization scenarios. Hence, all framework parameters were adjusted identically for all ten problems.

6.2.2 MA Versions and Parameterization of the Method

This section presents the parameter setting and the intermediate algorithms developed to reach the final version of the framework-based MA for single global optimization, previously described in Section 5.2. These versions were designed based on a constructive approach, starting from a more straightforward MA to the final method. They were created to aggregate and analyze different search components and to delineate the methods for the other optimization scenarios.

As presented in Section 5.2, the general MA framework (Algorithm 7), combined with the ABC algorithm (Algorithm 8), has a parameter set necessary for its execution. Some of these parameters have already had their values defined with the search components and do not change regarding the algorithm versions presented in this section. Thus, the proposed MA receives as input parameters the initial number of layers N_{layers} in the tree, the maximum number of individuals Max_NS in the entire population of the MA, the number of generations g_{core} executed by the core metaheuristic, and the maximum number of fitness function evaluations Max_Evls. It also receives as an input parameter, when necessary depending on the version, the period in terms of generations n_gen used to calculate the simple moving average (SMA). Thus, the input parameters of the method and their initial values are described as follows:

- Number of layers in the tree structure: the number of layers in the ternary tree structure also defines the number of nodes in the structure (Equation 5.1). We adopted an initial tree structure with 3 layers ($N_{layers} = 3$) as used in our previous works (CORRÊA; INOSTROZA-PONTA; DORN, 2017; CORRÊA et al., 2020; CORRÊA et al., 2016; CORRÊA et al., 2018; CORRÊA; DORN, 2020), and consequently the maximum of 13 nodes ($N_{nodes} = 13$). However, we observe that the number of layers is dynamically adapted over the process depending on the version;
- Maximum number of individuals in the entire population of the MA: this parameter also defines the maximum number of individuals in each node depending on the number of nodes (N_{nodes}) in the tree. We adopted $Max\_NS = 312$, and consequently the maximum of 24 individuals per node ($Max\_NS_{node} = 24$) due the number of nodes ($Max\_NS_{node} = Max\_NS/N_{nodes}$), following the parameterization of Corrêa and Dorn (CORRÊA; DORN, 2020);
- Number of generations executed by the core metaheuristic: this parameter represents the number of generations that the hybrid-ABC runs at each generation of the framework. g_{core} was defined as 10 to promote exploration intra-niche before the interactions between niches;
- Number of generations used to calculate the SMA: this parameter defines the period in terms of generations used to calculate the SMA in the control procedure for convergence and performance (Section 5.2.8). We adopted $n\_gen = 3$ to consider the exploration degree through a reasonable number of generations since $n\_gen = 3$ implies 30 generations of the core metaheuristic. This parameter is not used in all MA versions;
- Stop condition: the stop condition is defined by the number of fitness function evaluations performed throughout an execution. For all benchmark functions, we adopted a threshold of $Max\_Evls = 10^7$ fitness evaluations or when the algorithm reaches the 10-digit level of accuracy.

The set of input parameters of the MA framework and their initial values are summarized in Table 6.2.

To analyze the proposed MA framework for global optimization, this section details designed variants to discuss the modifications that occurred in the algorithm through-

Table 6.2: Summarization of the input parameters of the MA and their initial values

Parameter	**Initial value**
Number of tree's layers	$N_{layers} = 3$
Number of tree's nodes	$N_{nodes} = 13$
Maximum population size	$Max\_NS = 312$
Maximum node size	$Max\_NS_{node} = 24$
Inner generations of the core metaheuristic	$g_{core} = 10$
Complete generations used to calculate the SMA	$n\_gen = 3$
Stop condition	$Max\_Evls = 10^7$

Source: From the author (2022).

out the work's development. Thus, the main components and search strategies to be analyzed in the algorithm variants are described below:

- Hierarchical tree data structure (Section 5.2.1);
- Inner node recombination, selection and replacement strategies (Section 5.2.4);
- Interactions between nodes (Section 5.2.5);
- Local search strategy (LIP procedure, Section 5.2.6);
- Control procedure for convergence and performance (restarting and performance verification, Section 5.2.8);
- Dynamic Niche Size Procedure (DNSP, Section 5.2.9);
- Core metaheuristic (ABC algorithm, Section 5.2.12).

Therefore, six different variants were implemented considering the components above. We highlight that the discussion of these algorithms was done through an incremental process, i.e., the first version configures the baseline. The subsequent versions incorporate all the components added in the previous one. We note that the solutions for all methods are randomly initialized within the search space of each optimization function. The parameter values defined above (Table 6.2) remain the same whether the parameters are included in a given algorithm version. The last variant is the final MA framework for global optimization fully described in Section 5.2.

6.2.2.1 Version 1: Baseline algorithm

The first version was designed as the baseline algorithm for the MA framework and used as the basis for the following variants. The baseline follows the same structure of the MA for the PSP problem presented in the work of Corrêa and Dorn (CORRÊA;

DORN, 2020).

The baseline was implemented using the same hierarchical tree data structure described in Section 5.2.1. As mentioned, the tree structure provides a multi-population search strategy, which arranges the population in subpopulations into the tree's nodes. We note that for all versions of the method, the hierarchical tree was parameterized as a ternary tree according to our previous papers focused on the PSP problem (INOSTROZA-PONTA; FARFÁN; DORN, 2015; CORRÊA; INOSTROZA-PONTA; DORN, 2017; CORRÊA et al., 2020; CORRÊA et al., 2016; CORRÊA et al., 2018; CORRÊA; DORN, 2020).

In this version, the population is organized into a static tree structure with 3 layers ($N_{layers} = 3$) and 13 nodes ($N_{nodes} = 13$), according to the Equation 5.1. Each node has a static number of individuals of 24 solutions per node ($Max\_NS_{node} = 24$), which gives 312 solutions in total ($Max\_NS = 312$). The solutions are randomly initialized and divided into the tree's nodes without any niching strategy. Regarding the framework, the population size (NS) is also the maximum number of individuals allowed in the tree ($NS = Max\_NS$) and does not change over the process. Thus, each node represents a subpopulation of the MA with $M = 24$ solutions, which are then optimized based on the MA search mechanisms.

The nodes are optimized internally by the ABC core search metaheuristic, which is simpler than the final one (Section 5.2.12) and is further described. The interactions between nodes follow predefined hierarchical rules and are performed via recombination operations. Figure 5.1 shows the tree structure of the baseline algorithm with 3 layers, which cannot be dynamically adapted over the optimization process as the final version. After the subpopulations' initialization, the steps below are executed in every generation as described in Section 5.2.3, but with some differences.

- First step:

Each node runs the core metaheuristic independently (ABC algorithm, Section 4.5.1), where the search algorithm is executed by g_{core} generations in each node. Besides, each node performs g_{core} inner recombination operations with the solutions in its respective subpopulation. However, the offspring is inserted into the next population only if it is better than the worst solution and not by the Deterministic Crowding (DC) replacement strategy as in the final version (Section 5.2.4). We note that the replacement of the worst solutions increases selection pressure and speeds up the convergence, which may lead to premature convergence.

In this sense, the inner recombination aims to enhance the exploration of the search space of a given subpopulation. As in the final version, the operation is performed by the Uniform crossover, and the selection of individuals is made via a 3-way tournament selection strategy. As the proposed MA intends to provide a general optimization method, the inner recombination is used to complement the search mechanisms of the framework, favoring the exploration of intra-niche regions.

- Second step:

The interactions between nodes are performed following the hierarchical structure of the tree, with its neighbors from the same parent and with its parent (Figure 5.1). As in the final version, the interactions are performed by recombination operations using the Uniform crossover. However, this version only performs the bottom-up crossover operation (Algorithm 3, Section 5.2.5) and does not include the top-down operation. We note that the final version of the interactions between nodes is described in Section 5.2.5).

According to the Figure 5.1, the hierarchical sub-structure is divided into two levels: (*i*) upper level, which comprises a parent node with child nodes; and (*ii*) lower level, which includes the three child nodes of the parent node in the upper level. In the bottom-up crossover, the child nodes (lower level) do crossover operations with their neighbors, linked to the same parent (upper level), to generate three offspring for each group of neighbors (Algorithm 3, lines 2-10). One solution from each subpopulation is selected by ranking-based selection strategy as parents in each recombination operation. However, the offspring from the crossovers between the child nodes are integrated into the population of the respective parent only if they are better than the worst solutions, and not via the Restricted Tournament Selection (RTS) strategy as shown in Section 5.2.5.

- Third step:

In the baseline, the LS algorithm is also applied on the best solutions of each subpopulation via the LIP, described in Section 5.2.6. The algorithm incorporates the same LIP procedure of the final version as a fundamental part of the MAs. It has the continuous LS strategy, where the MA framework was combined with the Solis and Wets (SW) algorithm (SOLIS; WETS, 1981) as its LS strategy. The LS chain is used to control the SW parameters for each solution by keeping the historical memory of the LS procedures already performed on each individual. As previously discussed, such a strategy aims to adjust the exploitation intensity (number of fitness evaluations) applied to the MA throughout the evolutionary process.

Nonetheless, the only difference in this version is that the LIP is applied to one of all solutions of each subpopulation and not to only one of the $p_{best}\%$ best solutions. Such constraint was implemented in the following versions to focus the refinements only on the promising regions of the search space, giving to the exploration the role of improving the non-promising solutions. The LS restarting is also employed but considering the entire population to decide whether it will be executed. We note that we adopted the same parameterization of the LIP's final version (Section 5.2.6) for all intermediate variants.

- Fourth step:

As a central concern of the MA, population diversification is required to ensure a suitable algorithm's exploration and the ability to overcome local optima. Contrarily, the loss of population diversity may be necessary for the exploitation and convergence of the algorithm (SER et al., 2019; ČREPINŠEK; LIU; MERNIK, 2013). Thus, this version also implements a control mechanism for convergence but is simpler than the final one, detailed in Section 5.2.8. The baseline does not present any niching strategy at the initialization step, except for the hierarchical tree structure that segments the solutions in distinct subpopulations. As mentioned, it does not include the tree resizing procedure (TRP) to change the tree structure dynamically during the optimization process (Section 5.2.7). However, the control mechanism for convergence aims to ensure a certain degree of diversity whereas improving the best solutions in a simplest manner. With this, the baseline only uses the restarting procedure described in Section 5.2.8.

Therefore, it uses the coefficient of variation (CV) measure (Equation 2.6), discussed in Section 2.5, as the indicator of population diversity instead of the diversity measures used in the final version. So we note that this version does not employ the combination of diversity measures used to quantify the similarity and distribution of individuals, described in Section 5.2.8. The CV was employed as a first trial to evaluate the relationship between the population diversity and the Expr/Expt balance.

Like the final method, the restarting procedure is responsible for monitoring and independently restarting each subpopulation of the tree if it reaches a premature convergence according to the CV values. In the baseline, the subpopulation of a given node is reinitialized if it has reached a CV value less than $p\_expr\%$. CV values are expressed as a percentage in the continuous range $[0, 100]$, and the parameter $p\_expr\%$ is the threshold used to indicate whether the subpopulation has signs of convergence or not, as lower values of CV indicate loss of diversity and low algorithm's exploration. This version does not include the SMA as a second threshold for restarting.

Thus, for each $node_i (i = 1, \cdots, N_{nodes})$ in the tree structure, its subpopulation is reinitialized if:

$CV_{curr_i} < p_{expr}\%$, which indicates that the current CV value is less than $p_{expr}\%$.

If restarting, the procedure discards the entire subpopulation and generates a new one, only keeping the best solution. We observe that this version does not employ the performance verification procedure to control the performance of the MA by analyzing the number of improvements of solutions throughout a certain period of generations (Section 5.2.8). We adopted $p\_expr\% = 15\%$ as used in our previous work (CORRÊA; DORN, 2020).

- Fifth step:

Finally, we highlight that the method does not dynamically change the niche size nor reorganize the solutions into the nodes through the clustering-based Speciation algorithm (Section 5.2.9). So the DNSP was not considered in this version. The components not included in the baseline are described in the subsequent versions to form the final method.

- Core metaheuristic:

The core metaheuristic is responsible for the MA subpopulations' inner exploration (Section 5.2.12). Thus, the MA was combined with a modified version of the ABC algorithm (KARABOGA; BASTURK, 2007) since the first version. The baseline ABC was inspired by the one applied to the PSP problem, presented by Corrêa and Dorn (CORRÊA; DORN, 2020). It is a simpler version of the hybrid ABC algorithm detailed in Section 5.2.12. According to the literature (KARABOGA; BASTURK, 2007; AKAY; KARABOGA, 2012), the synergy of concepts between the MA and ABC algorithms seems suitable for the optimization domain understudy in the sense of efficient search space Expr/Expt. For that reason, we decided to adopt the ABC since the baseline variant.

This baseline version does not include all components of the final method shown in Algorithm 8 of Section 5.2.12. It follows the same optimization steps of the standard ABC (Algorithm 1, Section 4.5.1), which encompass the employed bees' stage, onlooker bees' stage, and the scout bees' stage. Such steps aim to simulate the foraging task performed by specialized bees of a honeybee.

Moreover, the algorithm does not implement the population division strategy, which splits each subpopulation into two groups according to the solutions' fitness quality to promote Expr/Expt. However, the individuals are mutated by the gbest search equation

(Equation 4.2) of the GABC (ZHU; KWONG, 2010) instead of the standard ABC mutation equation (Equation 4.1), previously explained in Sections 4.5.2 and 5.2.12. So the entire population is guided by the best individual of the subpopulation to move toward the global of this niche and at least increase the convergence rate, refining the accuracy of individuals.

As previously discussed in Sections 4.5.1, 4.5.2 and 5.2.12, the ABC (Algorithm 1) has the ability of search space exploration but presents some inefficiencies of exploitation, which can imply in slow convergence rate and stagnation (ZHU; KWONG, 2010; AKAY; KARABOGA, 2012; LI; NIU; XIAO, 2012). Thus, we adopted the mutation rate-MR strategy (AKAY; KARABOGA, 2012) to overcome the slow convergence of the standard algorithm due to the mutation of only one dimension of the individual at each operation. The employed and onlooker bees' phases consider more dimensions to be mutated at each operation based on the defined threshold MR. We employed rate-MR with $MR = 0.4$ by the suggestion of Akay and Karaboga (AKAY; KARABOGA, 2012) for continuous optimization, and we kept this strategy until the final version (Section 5.2.12).

Regarding the control parameters, the population size SN is defined by the subpopulation size submitted to it, and the MR parameter was defined above. The 'limit" parameter l for scout bees was initially set as the standard ABC $l = 200$ instead of the final version's $l = SN \times D$. The parameter l is the threshold that defines when an employed bee abandons the food source and becomes a scout bee, which means that if in over 200 generations a solution has not been improved, it is discarded. The scaling factors of the gbest equation δ and γ were defined as follows: (*i*) δ can assume a random real number in the range $[-1, 1]$; and (*ii*) γ is a random real number in the range $[0, 1.5]$. We highlight that this ABC version does not adopt any self-adaptive control parameter. Thus, random values of δ and γ are assigned to each variable of the solution being mutated.

6.2.2.2 Version 2

The second version was created from the baseline to test the frequency of the restarting procedure (Section 5.2.8). Thus, we slightly reduced the parameter $p\_expr\%$ in order to reduce the number of restarts performed throughout the optimization process. With this, we adopted $p\_expr\% = 5\%$ due to the high number of restarts performed by the first version. A high number of restarts over an optimization process may imply a slow convergence rate at the last generations of the process. On the contrary, the loss of population diversity in a given optimization period is necessary for the exploitation and convergence of the algorithm (ČREPINŠEK; LIU; MERNIK, 2013). So we used a lower $p\_expr\%$ value in an attempt to reach a reasonable convergence at the end of the process since the MA has other mechanisms to maintain the diversity at the beginning of the search.

Nonetheless, we also modified the inner recombination strategy (Section 5.2.4), where the offspring from the recombination is now inserted in the next subpopulation via a greedy selection between the new solution and its parents. The competition among the offspring and its parents decreases selection pressure and mainly supports diversity in the initial steps.

Therefore, we decreased the $p\_expr\%$ threshold to avoid high diversity levels at the last generations of the process but also changed the inner recombination procedure to not accelerate the convergence at the first generations with the replacement of the worst solutions.

6.2.2.3 Version 3

The third version received one of the main strategies focused on multimodal landscapes implemented in the framework. We adopted a cluster-based niching strategy (Section 5.2.2) to divide the entire population into subpopulations based on the spatial positions of the individuals. The clustering-based Speciation algorithm strategy was used to enhance the tree structure's potential to segment the population into distinct nodes.

Thus, the niching strategy is a distance-based neighborhood algorithm that divides the NS individuals of population P into NCl subpopulations, each of which comprises M individuals with adjacent locations in the search space. Such a strategy aims to maintain a diverse population longer over the process, preventing a premature convergence to local optima. We note that it was detailed in Algorithm 2 of Section 5.2.2.

From the inner recombination replacement strategy of the previous versions, in this variant, we adopted the DC replacement instead of the replacement by parents strategy (Section 5.2.4). The DC has the potential to maintain diversity by exploring the similarities between solutions. Thus, the DC has the same purpose as the other replacement component but may keep the diversity intra-niche more efficiently considering the distance between solutions. It was used to support the cluster-based niching strategy and enhance the balance between Expr/Expt.

Lastly, regarding the restarting procedure (Section 5.2.8), we adopted a different criterion in place of the CV measure of the first versions. The new criterion was employed as a first attempt to evaluate the balance between Expr/Expt by considering the distinct diversity measures detailed in Sections 2.5 and 5.2.8. Thus, as diversity measures are problem-dependent, we employed a strategy that combines the diversities measured in both the objective and decision spaces to quantify the search space, individuals' distribution, and fitness. The four distinct diversity measures, previously presented in Section 2.5, consist of two distance-based measures: (*i*) the dimension-wise diversity measure ($Div_{dimension}$, Equation 2.1); and (*ii*) the population center measure (Div^2_{center}, Equation 2.3). And two other measures focused on the objective space: (*iii*) the entropy-based measure ($Div_{entropy}$, Equation 2.4); and (*iv*) the fitness-based distance ($Div_{fitness}$, Equation 2.5). The usage of different measures was done to better evaluate the diversity oscillation throughout the search process. Besides, this threshold was included to support the DC replacement strategy during the execution of the core metaheuristic and inner recombination operations.

The restarting criterion uses as threshold ($p_{expr}\%$) the half of the maximum diversity values obtained during the process (Equation 6.13). Thus, the procedure evaluates separately whether the current value of each measure is less than its threshold. Hence, the subpopulation of a node is reinitialized if all current diversity values satisfy the threshold. This criterion may indicate whether the subpopulation has lost diversity and presents consistent signs of convergence when compared with different measures.

Therefore, for each $node_i (i = 1, \cdots, N_{nodes})$ in the tree structure, its subpopulation is reinitialized wether the condition below is satisfied for each diversity measure Div_j ($j \in 1, 2, 3, 4$):

$Div_{curr_{ij}} < p_{expr_{ij}}\%$, which indicates that the current value of the j-th measure is less than the half of its maximum value, given by Equation 6.13.

$$p_{expr_{ij}}\% = \left(\frac{Div_{max_{ij}}}{2}\right) \tag{6.13}$$

Where $p_{expr_{ij}}\%$ is the threshold for the j-th measure related to the i-th node.

6.2.2.4 Version 4

In this version, we enhanced the restarting procedure (Section 5.2.8), changing its criterion again. The same diversity measures used in the previous version were combined to form a single measure in an attempt to better address the Expr/Expt ratio over the optimization. The ensemble of measures replaced the values of each measure evaluated individually by the average of the consolidated values of the four measures. This combination of measures was adopted to favor their strengths and overcome their shortcomings to distinguish distinct populations over the search space and provide better guidance for the process. With this, the single measure incorporated in the MA denotes the average of the normalized values of the four diversity measures considered ($Div_{combined}$, Equation 5.4). We note that this measure was explained in Section 5.2.8.

Moreover, to quantify the Expr/Expt balance using the combined diversity measure ($Div_{combined}$), we introduced a definition to represent it as the percentage of Expr/Expt performed by a given algorithm (Section 2.6). Considering the combined diversity measure ($Div_{combined}$, Equation 5.4 of Section 5.2.8), the exploration ($Expr\%$, Equation 5.5) and exploitation ($Expt\%$, Equation 5.6) ratio is defined by the relative diversity of the population. The relative diversity was detailed in Section 5.2.8 and represents the ration between the diversity rate of the current iteration and the initial diversity value calculated from the first initialized population. Thus, the percentage of exploration ($Expr\%$) describes the exploration effort as the relationship between the diversity in each iteration and the initial diversity rate. The percentage of exploitation ($Expt\%$) is calculated as the complementary percentage of $Expr\%$ and describes the exploitation effort. As $Expr\%$ and $Expt\%$ are mutually complementary, we focused on the $Expr\%$ values to address both efforts.

Therefore, the niche restarting procedure evaluates each tree's subpopulation in terms of the $Expr\%$ values (Equation 5.5) to monitor an eventual premature convergence. With this, a node's population is reinitialized if it has reached $Expr\%$ less than $p_{expr}\%$. $Expr\%$ is expressed as a percentage in the continuous range $[0, 1]$, and the parameter $p_{expr}\%$ is the threshold used to indicate whether the subpopulation shows a tendency of convergence or not. Thus, for each $node_i (i = 1, \cdots, N_{nodes})$ in the tree structure, its subpopulation is reinitialized if:

$Expr_{curr_i} < p_{expr}\%$, which indicates that the current $Expr\%$ is less than $p_{expr}\%$;

If restarting, the procedure discards the entire subpopulation and generates a new one, only keeping the best solution. So, restarting is used to diversify the population. However, depending on the value adopted for the threshold $p_{expr}\%$, the $Expr\%$ may indicate the convergence of the population to a local optimum. Thus, recommended values for $p_{expr}\%$ are in the continuous range $[0, 0.1]$ (MORALES-CASTAÑEDA et al., 2020). Hence, we used $p_{expr}\% = 0.01$, which means that the restarting is performed in a given niche if the exploration rate is less than 1%. We note that it was kept until the final version (Section 5.2.8).

Moreover, suppose the restarting procedure is applied to any subpopulation. In that case, the clustering-based Speciation algorithm is performed again to reorganize the most similar solutions in a single node according to their similarities (Section 5.2.9). We introduced this strategy to focus the search on distinct regions and avoid losing the achieved regionalization.

Regarding the ABC metaheuristic from the previous versions, we incorporated the population division strategy, which splits each ABC subpopulation into two groups according to the individuals' fitness quality. This schema is detailed in Section 5.2.12. At the beginning of each generation, the individuals of a node are classified as superior (better fitness) or inferior individuals (poorer fitness). To divide the solutions, we adopted the midpoint of the set of fitness values (md) as fitness division threshold (Equation 5.11 of Section 5.2.12). Then, solutions of a given niche with similar fitness may focus on exploration or exploitation and not on both based on the split of the population. The size of the groups varies according to the optimization status of the subpopulation. Such a strategy aims to balance the Expr/Expt mechanisms of the ABC by avoiding using a single mutation for solutions in different stages of convergence.

The superior individuals are mutated by the gbest search equation (Equation 4.2) of the GABC to accelerate the population convergence. The inferior ones should explore other regions of the search space and preserve population diversity. Contrarily, the inferior ones are updated by the mutation search equation "DE/rand/1" from the DE algorithm (Equation 4.4 of Section 4.5.3). "DE/rand/1" is a random mutation with no bias to any search direction, which may favor the exploration and preserve population diversity.

Moreover, in this version, we introduced a self-adapting control parameter mechanism for the search equations' scaling factors δ and γ of the gbest and F of the "DE/rand/1" equation. It was inspired in the jDE algorithm (BREST et al., 2006; BREST; MAUČEC; BOŠKOVIĆ, 2019), previously explained in Section 4.5.3. With this, each solution of the subpopulation has its control parameters δ, γ, and F, which are evolved together with the objective variables throughout the optimization. Hence, the scaling factors are kept the same for a given candidate solution during the update operation and do not change regarding each mutant variable, as in the previous versions. This modification was done due to the self-adaptive strategy adopted for the scaling parameters. We note that the adaptive schema is detailed in Section 5.2.12, where new parameter values for δ, γ and F are calculated before the update operation of a given solution, as shown in the the Equations 5.12, 5.13 and 5.14, respectively. Also, the parameters are initialized at the beginning of the

MA execution in the initialization step of the framework.

Lastly, regarding the “limit” parameter l for the threshold of discarding the scout bees in the ABC, we adopted the proportion $l = SN \times D$ instead of the constant $l = 200$ of the previous versions. SN is the population size, and D represents the problem dimension. According to Akay and Karaboga (AKAY; KARABOGA, 2012), this proportion fits well with dynamic population sizes and varied problem dimensions.

6.2.2.5 Version 5

This version mainly incorporated the dynamic procedures regarding the hierarchical tree structure of the framework. The first component is the TRP, previously explained in Section 5.2.7. It consists of a deterministic algorithm that shrinks the number of layers in the tree linearly according to the number of fitness function evaluations. The TRP aims to enforce the algorithm’s exploration at the beginning of the process and reach an acceptable convergence at the last generations. The strategy removes one layer of the tree at every rm_{layer} number of fitness function evaluations of the framework. So given the input parameters, rm_{layer} is the ratio between $Max\_Evls = 10^7$ and $N\_initial_{layers} = 3$ (Equation 5.3). In this scheme, the outermost layer of the tree is removed. The MA will have precisely one layer with a single node at the end of the optimization process.

Thus, we highlight that the TRP combined with the clustering-based niching strategy promotes population diversity at the early optimization stages. On the other hand, it reduces the tree structure over the optimization to increasingly focus on the most promising regions found.

We also enhanced the niche restarting procedure from the previous one in this version. Besides the exploration effort ($Expr\%$) criterion calculated from the combined diversity measure ($Div_{combined}$), we introduced a second criterion to follow the trend of variation of the $Expr\%$ and support the decision making.

Therefore, the population of a given node is reinitialized if it has reached $Expr\%$ less than $p_{expr}\%$ and if the SMA of the $Expr\%$ values shows a decreasing tendency when compared with the SMA calculated in the previous period of generations. As already detailed in Section 5.2.8, the SMA is used to analyze the $Expr\%$ values by creating two comparison periods of averages. These periods encompass a given number of overlapped generations, where the latest period may drop the earliest $Expr\%$ value (oldest generation) and includes the latest $Expr\%$ (current generation). Hence, the SMA of the current and the past periods are compared to verify the loss of diversity in a given niche, which, combined with the low exploration rate, corroborates with the convergence of the subpopulation.

Thus, for each $node_i(i = 1, \cdots, N_{nodes})$ in the tree structure, its subpopulation is reinitialized if:

1. $Expr_{curr_i} < p_{expr}\%$, which indicates that the current $Expr\%$ is less than $p_{expr}\%$;
2. $SMA\_Expr_{curr_i} < SMA\_Expr_{previous_i}$, which indicates that the current SMA of $Expr_i\%$ ($SMA\_Expr_{curr_i}$) values is less than the SMA of n_gen generations ago ($SMA\_Expr_{previous_i}$).

Following the previous versions, if restarting, the procedure discards the entire subpopulation and generates a new one, only keeping the best solution. We observe that this is the final version of the restarting strategy. The complete procedure is shown in Algorithm 4 of Section 5.2.8.

As mentioned in the baseline version, we first adopted a simplest version of the control procedure for convergence and performance and then enhanced it. Thus, in this version, we incorporated the second procedure of this strategy, the performance verification, described in Section 5.2.8. The procedure aims to control the algorithm's performance by analyzing the number of improvements of solutions throughout a given period of generations. Similar to the restarting procedure, but less aggressively, it prevents the method's stagnation by dynamically changing the size of each niche depending on the exploration status. The procedure works using a dynamic threshold ($evol_{time}$) related to the period with no improvement (not_imp) of the gbest solution.

The parameter not_imp is initialized with 0 and incremented by 1 at each consecutive generation that the gbest solution is not improved. Then, if the gbest solution has not been improved during $evol_{time}$ number of generations ($not\_imp > evol_{time}$), this procedure applies the DNSP to the entire population of the MA. We observe that the DNSP was also implemented in this version and is responsible for changing the size of subpopulations considering the increasing/decreasing of the exploration given by the SMA of two periods of $Expr\%$ values. It also reorganizes the individuals through the clustering-based niching algorithm. DNSP is fully explained in Section 5.2.9.

To update the dynamic threshold $evol_{time}$, the difference between the number of current generation (gen_{curr}) and the previous threshold used in the last update of the procedure ($evol_{previous}$) is stored as the new value for the threshold ($evol_{time} = gen_{curr} - evol_{previous}$). With this, whenever the condition ($not\_imp > evol_{time}$) is satisfied, the threshold is updated as the number of generations that the MA took to reach the no improvement threshold since the last time that this condition was satisfied. Thus, the more improvements without satisfying the condition for the DNSP execution, the higher the threshold $evol_{time}$ when updated. The variables $evol_{time}$ and $evol_{previous}$ are initialized with the low value of 1 in an attempt to quickly adjust the size of subpopulations to suitable search indicators at the beginning of the process in case of no improvement of the best solution. The Algorithm 5 of Section 5.2.8 shows the pseudocode of the performance verification procedure.

We note that the complete procedure for convergence and performance is described in Section 5.2.8, encompassing the niche restarting and the performance verification procedures.

As a complement to the components already included in the method, we implemented the DNSP to dynamically adapt the size of the niches and then reorganize the individuals by the clustering-based niching algorithm. Besides, DNSP aims to better follow the optimization process in terms of the Expr/Expt efforts and the algorithm's performance by improving solutions. The complete procedure is described in Section 5.2.9.

The population size is one of the most critical parameters of a metaheuristic since it strongly influences the balance between Expr/Expt (SER et al., 2019). As mentioned, the population size parameter is problem-dependent; hence combining strategies to adapt it throughout the search process with other optimization issues, such as the Expr/Expt balance and the algorithm's ability to improve the best solutions, can enhance the final results (ALETI; MOSER, 2016; POLÁKOVÁ; BUJOK, 2018; SER et al., 2019).

Therefore, the DNSP changes the size of subpopulations according to the minimum (Min_NS_{node}) and maximum (Max_NS_{node}) number of individuals allowed in each node. The Min_NS_{node} is defined regarding the minimum number of solutions required in the core metaheuristic. In our case, we adopted $Min\_NS_{node} = 5$, which is suitable for most population-based metaheuristics, including the ABC and DE algorithms. The Max_NS_{node} (Equation 5.9 of Section 5.2.9) is defined based on the ratio between the parameters that represent the maximum number of individuals (Max_NS) allowed in the entire method's population and the number of nodes (N_{nodes}) in the tree structure. As previously described in Table 6.2, we adopted as initial values for the input parameters $Max\_NS = 312$ and $N_{layers} = 3$, which, consequently, gives 13 nodes ($N_{nodes} = 13$) at the beginning of the process. With this, the maximum number of individuals allowed in each node is 24 ($Max\_NS_{node} = 24$). However, we observe that the number of layers is dynamically adapted whenever the TRP is applied. Hence, the N_{nodes} in the tree decreases, and the Max_NS_{node} increases over the process.

The DNSP is applied whenever: (*i*) the TRP (Section 5.2.7); (*ii*) the LS restarting (Section 5.2.6); (*iii*) the niche restarting (Section 5.2.8); and (*iv*) the performance verification procedure (Section 5.2.8) are executed. All of these components are already included in this version. However, the conditions to change the subpopulation size in DNSP are different depending on the procedure executed.

The first scenario is related to the TRP. At the end of this procedure, we introduced the execution of the DNSP. Thus, the DNSP defines a new size of subpopulations (M) randomly based on the range of minimum (Min_NS_{node}) and maximum (Max_NS_{node}) number of individuals allowed in each node. The size of subpopulations is randomly defined in this scenario because whenever the TRP is executed, it changes the tree structure and modifies the maximum number of individuals allowed in each node. Then, any condition based on the exploration degree would be affected.

The second scenario is related to the execution of the LS restarting and the performance verification procedures. At the end of them, the DNSP dynamically changes the size of subpopulations of the tree, considering the variation in the increase and decrease of the exploration given by the SMA of two distinct periods of $Expr\%$ values. Thus, the DNSP adapts the niche size regarding two significant concerns of the optimization process, i.e., the exploration effort through the population diversity degree and the performance via the improvement of best solutions.

It changes the size of subpopulations based on the exploration status of the method in a given generation. The exploration status is given by the average $Expr\%$ of the entire MA population ($Expr_{avg}\%$, Equation 5.10 of Section 5.2.9). Then, we used the SMA to analyze the $Expr_{avg}\%$ values by creating two comparison periods of averages. Thus, if the current SMA of $Expr_{avg}\%$ is less than the previous period (exploration is decreasing), the size of each subpopulation is randomly increased considering the range between the current subpopulation size and the Max_NS_{node}. We note that increasing the number of solutions may increase the exploration in terms of population diversity. On the other hand, if the current SMA of $Expr_{avg}\%$ is higher than the previous period (exploration is increasing), the size of each subpopulation is randomly decreased by a random number generated in the range of the current subpopulation size and the Min_NS_{node}. Analogously to the first condition, reducing the subpopulation size may increase the exploitation in terms of population diversity.

The third scenario concerns the niche restarting procedure. Whenever the restarting procedure is executed, the DNSP is performed. However, in this case, the niche size is not changed as in the above scenarios, but only the clustering-based niching procedure is executed.

Therefore, after any of the above scenarios, the DNSP performs the clustering-based Speciation (Algorithm 2 of Section 5.2.2) to reorganize the individuals into the nodes according to their similarities.

6.2.2.6 Version 6: Final version

Version 6 encompasses all the other components previously presented and represents the MA's final version for global optimization, which is fully described in Section 5.2. This version received enhancements related to the synergy of the MA components over the algorithm's optimization. Thus, we implemented the execution and priority order needed to run the procedures, focusing on orchestrating the search mechanisms. The priority order of execution for the framework components is explained in Section 5.2.10.

Another modification in this version concerns the interactions between nodes (Section 5.2.5). As mentioned in the baseline, the MA was using only the bottom-up crossover operation (Algorithm 3, Section 5.2.5). So we included the top-down operations to promote the knowledge sharing inter-niches and population diversification throughout the evolutionary process.

Considering the division of the hierarchical sub-structure between upper and lower levels illustrated in Figure 5.1. In the top-down crossover, the parent nodes (upper level) perform crossover operations with their children (lower level) to generate three offspring (Algorithm 3, lines 12-23). Then, the resulting offspring from the crossover operation between each parent and its children are integrated into the respective child subpopulation.

We observe that different from the previous versions, in the final one, the offspring from both bottom-up and top-down crossover operations are integrated into the respective subpopulations by the RTS strategy (Section 4.6.1) and not via a greedy competition with the worst solutions. The RTS was used to prevent different solutions from competing with each other to maintain the intra-niche cohesion and the inter-niche diversity.

Moreover, such a combination of the top-down and bottom-up interactions may increase the MA's exploratory potential and the synergy between the other search components. The two types of recombination were designed to integrate all the nodes as a way of knowledge sharing inter-niches and population diversification. The information of each one can eventually be passed to all the nodes throughout the evolutionary process.

Finally, we modified the LIP (Section 5.2.6) to be applied only to one of the $p_{best}\%$ best solutions of each subpopulation and not to one of all solutions. It was implemented to enhance the exploitation of the promising regions of the search space or escape from local minima. We adopted $p_{best}\% = 25$, which refers to the 25% best solutions of each subpopulation.

6.2.2.7 Summarization of the MA Versions

The framework-based MA for continuous optimization was designed based on an incremental strategy, starting from a baseline version to the final method. We presented six variants that are detailed in the sections above. The complete MA is described in Section 5.2 and the Algorithm 7 shows its pseudocode.

Therefore, the intermediate versions were created to combine search components with distinct purposes focused on global optimization with multimodal objective functions. We highlight that the framework aims to be a general optimization method that can be easily adapted to other search strategies and components. This incremental design of the method was done to illustrate some of these possibilities. Table 6.3 summarizes the main components incorporated in each variant, the search strategy differences, and concepts related to them. As the development of the algorithms was done through a constructive process, we note that the following versions incorporate all the components added in the previous one.

6.2.3 Results and Discussion

6.2.4 MA Versions

To test the performance of the presented MA versions, we applied them to the 100-Digit Challenge on single objective real-parameter optimization, as previously mentioned. The benchmark functions represent ten minimization problems (Table 6.1), where the goal is to compute each function's minimum value to 10 digits of accuracy using only one algorithm. The evaluation criteria are detailed in Section 6.2.1.

Thus, the score for a given function is the average number of correct digits in the best 25 of 50 trials. The results of the 50 runs are sorted according to the final number of

Table 6.3: Summarization of the MA variants, search strategy differences and main concepts related to them

Version	Component	Search strategy differences	Search concept
V1	`Tree data structure`	Static	Divide population in nodes
	`Tree initialization`	Random	Normal initialization
	`Inner recombination`	Replacement of worst solutions	May cause premature convergence
	`Node interactions`	Only bottom-up operation	Limit the knowledge sharing
	`Node interactions`	Replacement of worst solutions	May cause premature convergence
	`LIP`	Consider all subpopulation to LS	Does not focus on promising regions
	`TRP`	-	-
	`Control mechanism`	Restarting with CV criterion	Diversification with $p\_expr\% = 15\%$
	`DNSP`	-	-
	`Priority order`	-	-
	`ABC mutation`	Only gbest equation	Increase niche convergence
	`ABC pop. division`	-	-
	`ABC adaptive factors`	-	-
	`ABC limit parameter`	$l = 200$	Recommended but static
V2	`Control mechanism`	Restarting with CV criterion	$p\_expr\% = 5\%$; Regulate diversity
	`Inner recombination`	Replacement with parents	Decrease selection pressure
V3	`Tree initialization`	Clustering-based Speciation	Niching
	`Inner recombination`	DC replacement	Diversity preservation
	`Control mechanism`	Restarting with 4 measures	Follow diversity measures
V4	`Control mechanism`	Restarting with combined measure	Population diversification
	`DNSP`	Only re-clustering	Niching
	`ABC mutation`	gbest and “DE/rand/1”	Expr/Expt balance
	`ABC pop. division`	Based on fitness	Expr/Expt balance
	`ABC adaptive factors`	self-adapting strategy	Adaptation to the problem
	`ABC limit parameter`	$l = SN \times D$	Fit pop. size and problem dimension
V5	`TRP`	Linear reduction of tree nodes	Expr/Expt balance
	`Control mechanism`	Restarting with combined measure	Diversification (SMA of Expr)
	`Control mechanism`	Performance verification	Expr/Expt balance
	`DNSP`	Adaptive niche size; Re-clustering	Expr/Expt balance; Overcome issues
V6	`Node interactions`	Bottom-up and top-down operations	Increase MA’s exploratory potential
	`LIP`	Consider the 25% best solutions to LS	Focus on promising regions
	`Priority order`	Regulate components’ execution	Promote the components’ synergy

Source: From the author (2022).

correct digits that each trial found (best fitness values). Each run can score up to 10 points (correct digits) for a function. Moreover, the best 25 runs can score up to 250 points. Then, the score for this function is the average number of correct digits found in the best 25 runs. The total score of the algorithm is calculated as the sum of the scores for all 10 functions.

This section presents only the total score of each version obtained from all 10 functions’ scores for the best 25 of 50 runs. In the next section, we discussed the performance of the final version regarding each benchmark function and compared with the most relevant methods in the field, according to the analysis of challenge results of CEC 2019, GECCO 2019, and SEMCCO 2019 (PRICE et al., 2019). So the obtained total score of each version is described in Table 6.4. Besides, Figure 6.8 shows the bar plot of these results.

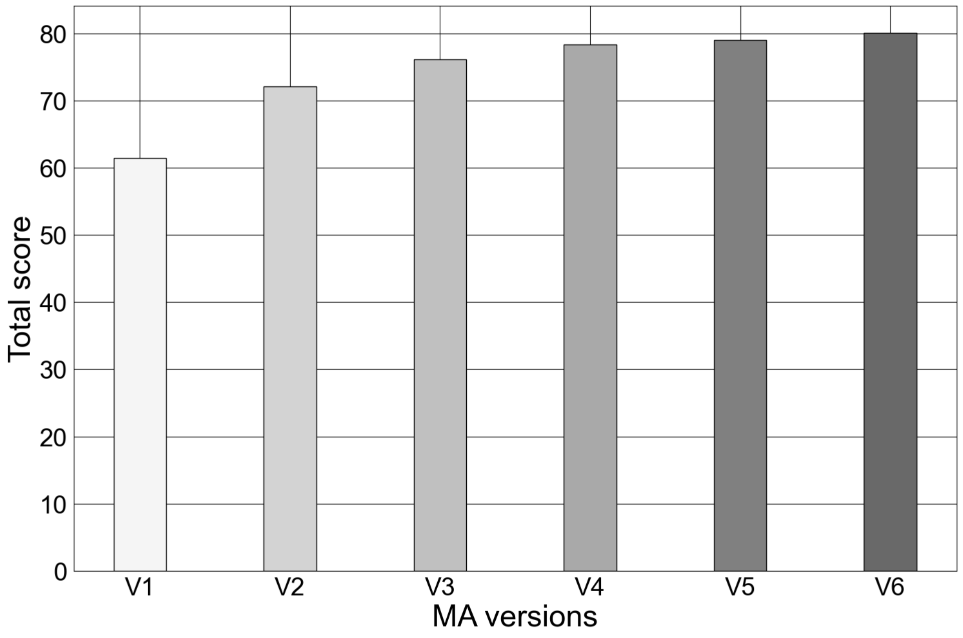
According to the results of Table 6.4, we observe that the incremental design of the

Table 6.4: Total score obtained from all 10 functions for the best 25 of 50 runs of each MA version

MA version	Total score
Version 1	61.48
Version 2	72.16
Version 3	76.12
Version 4	78.4
Version 5	79.0
Version 6	**80.12**

Source: From the author (2022).

Figure 6.8: Representation of the total score obtained from all 10 functions for the best 25 of 50 runs of each MA version



Source: From the author (2022).

MA was able to improve the final result of the last version. These results could indicate that the final version of the method, including all of the described components, has the potential to improve the results when dealing with global optimization problems with multimodal objective functions.

The proposed versions were implemented following the points of interest outlined for this thesis to reach a final framework-based MA for multimodal optimization. Thus, from the results illustrated in Table 6.4, we note that the modifications made throughout the method's development regarding Expr/Expt balance, improvement of solutions, and parameter control were able to support the MA in the optimization process. Nonetheless, it is observed that the MA still needs improvements to enhance the obtained results, mainly concerning the orchestration of all the components implemented over the algorithm's optimization.

In the following section, we detail the obtained results of the final version of the method regarding each of the benchmark functions. In addition, we conducted a convergence analysis regarding the best fitness values obtained throughout the experiments. Also, we illustrate the variability of the method in terms of Expr/Expt, its potential to handle the trade-off and improve the subpopulation solutions.

6.2.5 Final Method for Global Optimization

The obtained results of the MA's final version are described in Table 6.5. The score for a given function is the average number of correct digits in the best 25 of 50 trials, which is indicated in the last column. The results of the 50 runs are sorted according to the final number of correct digits that each trial found (best fitness values). Each run can score up to 10 points (correct digits) for a function. Furthermore, the best 25 runs can score up to 250 points. Then, the score for this function is the average number of correct digits found in the best 25 runs. The table shows the final number of correct digits for each trial, making the sum for a row equal to 50. For instance, regarding the obtained results for function F9, the method has reached 9 digits of accuracy (9 points) in 4 runs, 8 digits in 19 runs and 7 digits of accuracy in 24 runs, which encompass the best 25 runs. Thus, the method obtained 202 points considering the best 25 runs, which resulted in an average score of 8.08 (202/25). The total score is calculated as the sum of the scores for all 10 functions, and it is presented in the last column.

According to the results of Table 6.5, the proposed MA has reached a perfect score for 7 out of 10 functions. The total score achieved was 80.12 for 50 runs of all benchmark functions. Thus, to situate our results according to the most relevant methods in the field, we refer to the report of Price et al. (PRICE et al., 2019) concerning the analysis of competition results. The report combines the challenge results of CEC 2019, GECCO 2019, and SEMCCO 2019. It has classified the methods into two groups: (*i*) the primary algorithms, which are the tuned methods for specific functions; and (*ii*) the secondary algorithms, with most being among the better-performing algorithms at prior, time-limited competitions. In general, the secondary algorithms were not tuned, but run with previously published control parameter defaults. So 18 primary and 20 secondary

Table 6.5: Fifty runs for each function sorted by the number of correct digits

Function ID	**Number of correct digits**											**Score**
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
`F1`	0	0	0	1	0	4	1	0	0	0	44	10
`F2`	49	1	0	0	0	0	0	0	0	0	0	0.04
`F3`	0	0	0	0	0	0	0	0	0	0	50	10
`F4`	0	0	0	0	0	0	0	0	0	0	50	10
`F5`	0	0	0	2	0	0	0	0	0	0	48	10
`F6`	0	0	0	0	0	0	0	0	0	0	50	10
`F7`	0	0	5	0	0	0	0	0	0	0	45	10
`F8`	0	0	50	0	0	0	0	0	0	0	0	2.0
`F9`	0	0	0	0	0	0	3	24	19	4	0	8.08
`F10`	0	0	0	0	0	0	0	0	0	0	50	10
											Total score:	**80.12**

Source: From the author (2022).

algorithms were reported, totaling 38 methods in the competition.

According to the report results, only the jDE100 algorithm (BREST; MAUČEC; BOŠKOVIĆ, 2019) was able to reach the perfect score of 100, and hence was the winner of the competition (PRICE et al., 2019). We note that jDE100 was previously described in Section 4.5.3. The average score of the challenge among primary algorithms was 80.11, whereas the average score among secondary methods was 52.47. However, as expected, the algorithms that were not tuned tended to perform worst, with a lower score than the primary algorithms. For instance, the best score among secondary algorithms was 75.72, while the canonical ABC algorithm obtained the worst score of 9.64 and could not find 10 digits for any function. As mentioned earlier, our method was not tuned to any specific function to evaluate its robustness given a set of distinct benchmark functions. With this, the total score of our method would be 11th place among the 38 methods. However, we highlight the efforts to design the intermediate versions of the MA used to delineate the final version.

Furthermore, we observe that the proposed MA could find only one correct digit in function F2 and has scored 0.04, which may indicate some problem in our implementation since F2 was classified as an intermediate function in terms of difficulty. Regarding the report, the difficulty degree of functions was defined based on the average number of correct digits found by the 38 algorithms. The rank for functions from easy (1) to hard (10) is shown in Table 6.6.

Therefore, the results of the MA framework described in Table 6.5 indicate that

Table 6.6: Rank for the difficulty degree of functions from easy (1) to hard (10) based on the average number of correct digits found by the 38 algorithms

Difficulty degree	**Function ID**
1 (easiest)	F6 (9.2136 digits)
2	F5 (9.1726 digits)
3	F10 (8.6231 digits)
4	F1 (8.6126 digits)
5	F2 (7.6368 digits)
6	F3 (7.4389 digits)
7	F4 (6.7578 digits)
8	F7 (3.4421 digits)
9	F9 (2.7389 digits)
10 (hardest)	F8 (1.9268 digits)

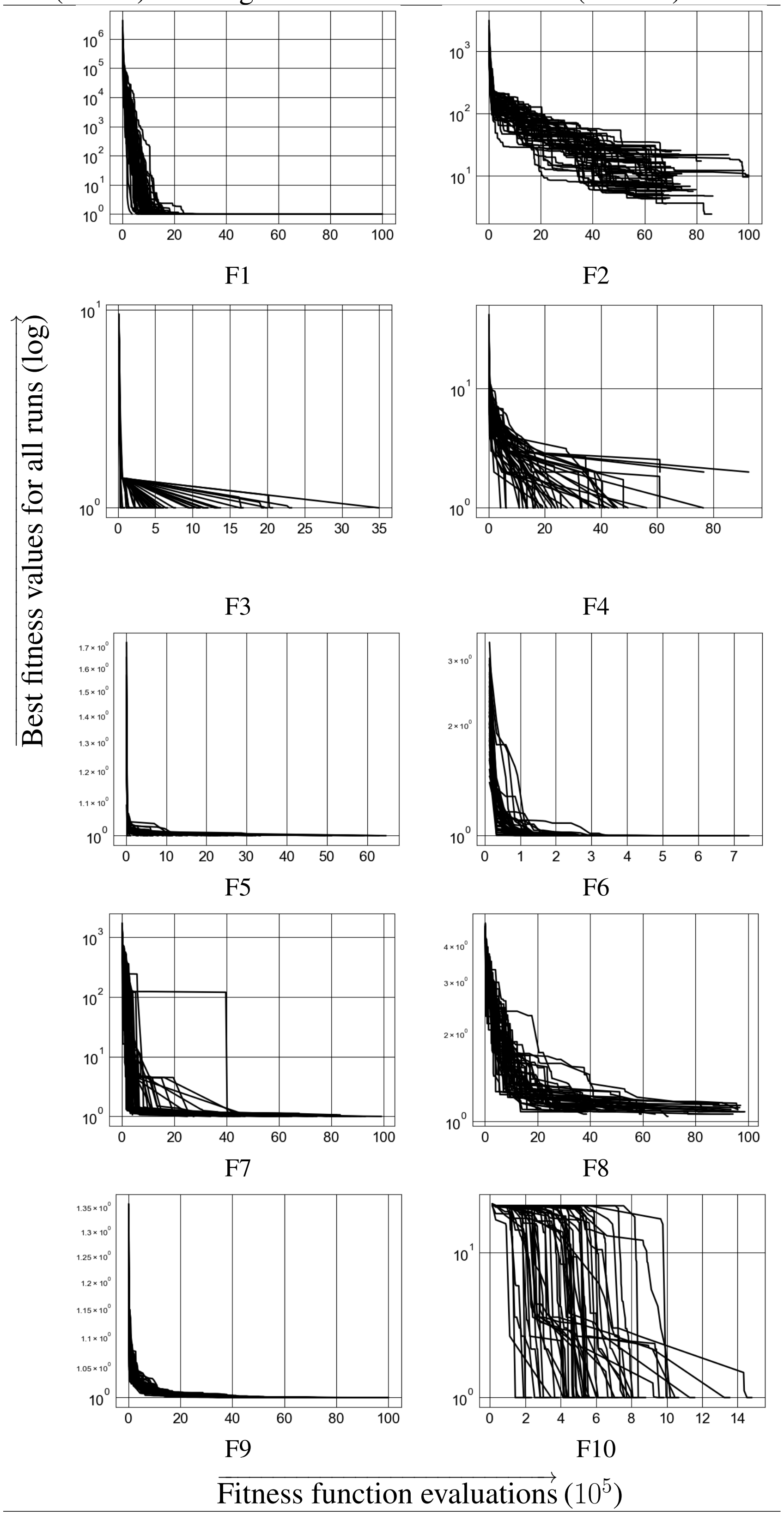
Source: From the author (2022).

our method was able to reach reasonable results for almost all functions, including the hardest ones, except for F2 and F8. Nevertheless, these results demonstrate the importance of adapting the method to the optimization function through efficient exploration of the search space, the proper ratio between Expr/Expt mechanisms, and control parameter strategies to adjust the search components based on the optimization process.

Figure 6.9 shows the convergence plots for each benchmark function (F1-F10) (Table 6.1). It illustrates the convergence of the method considering the best fitness values throughout each of the 50 runs. Moreover, to illustrate the convergence of the MA considering each node of the tree, similar to the aforementioned plots, Figure 6.10 shows the algorithm's convergence regarding the best fitness values of each node of the tree throughout each of the 50 runs. We note that each color in the plots represents the nodes of each layer of the tree. For both figures, it is observed that the range of x-axis in plots, i.e., the number of fitness function evaluations performed in each run, may vary due to the stop criterion adopted, where the method's execution terminates whether the algorithm reaches the 10-digit level of accuracy or regarding the maximum number of function evaluations allowed per execution ($Max\_Evls = 10^7$). Also, the y-axis values are expressed in the logarithm scale but preserve the minimum value of 1 as the global minimum.

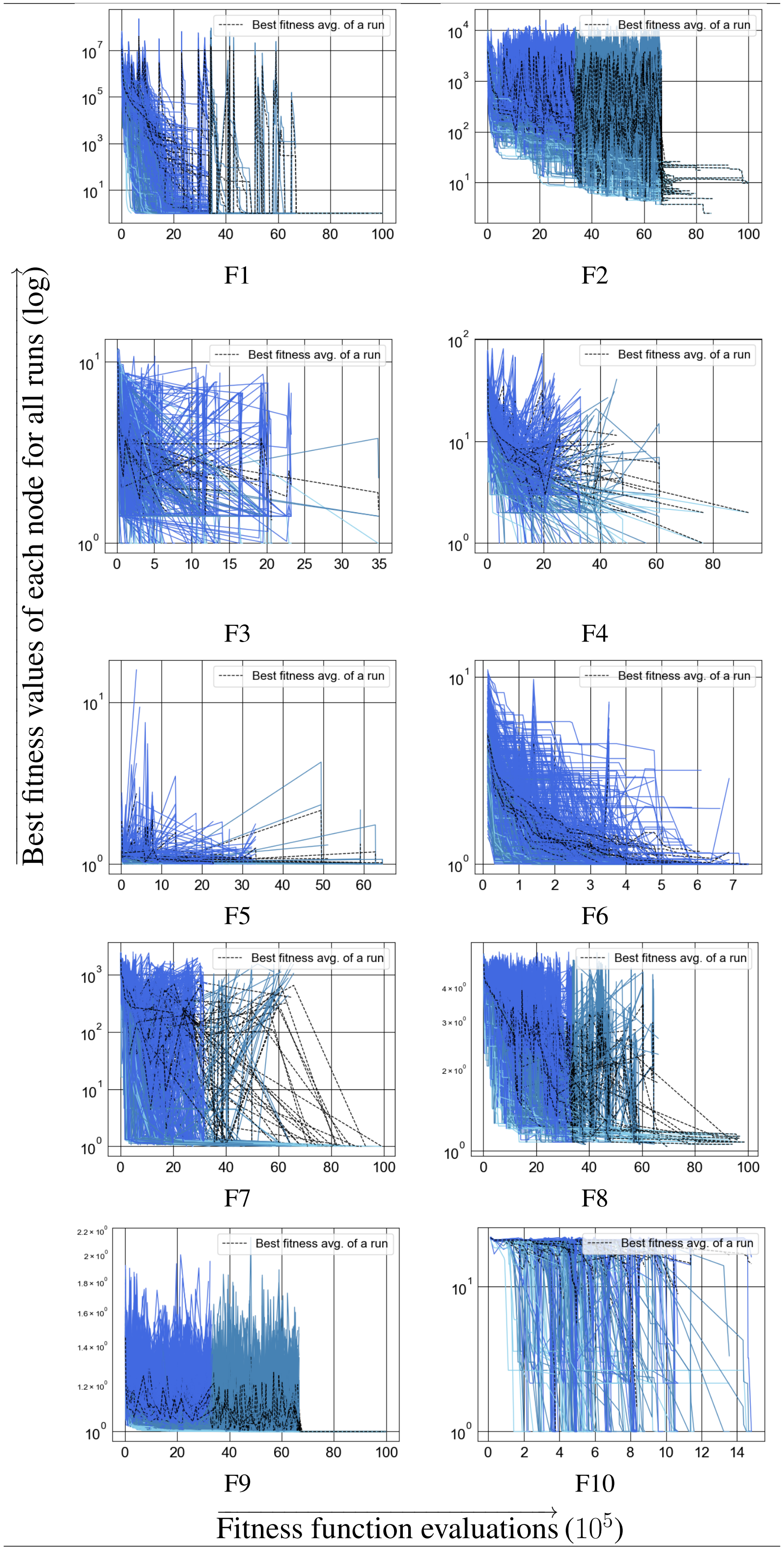
Analyzing the convergence plots in Figure 6.9, it is possible to note that the MA demonstrated a consistent convergence degree among the final values of each execution for almost all functions, except for F2. It corroborates the tendency of the method to reach similar numbers of correct digits in all executions of a given function, as shown in Table 6.5. Besides that, Figure 6.10 shows the convergence behavior of each node

Figure 6.9: Convergence of the MA regarding the best fitness values (**y-axis**) throughout each of 50 runs (**x-axis**) for the global benchmark functions (F1-F10)



Source: From the author (2022).

Figure 6.10: Convergence of the MA regarding the best fitness values of each node of the tree (**y-axis**) throughout each of the 50 runs (**x-axis**) for the global benchmark functions (F1-F10)



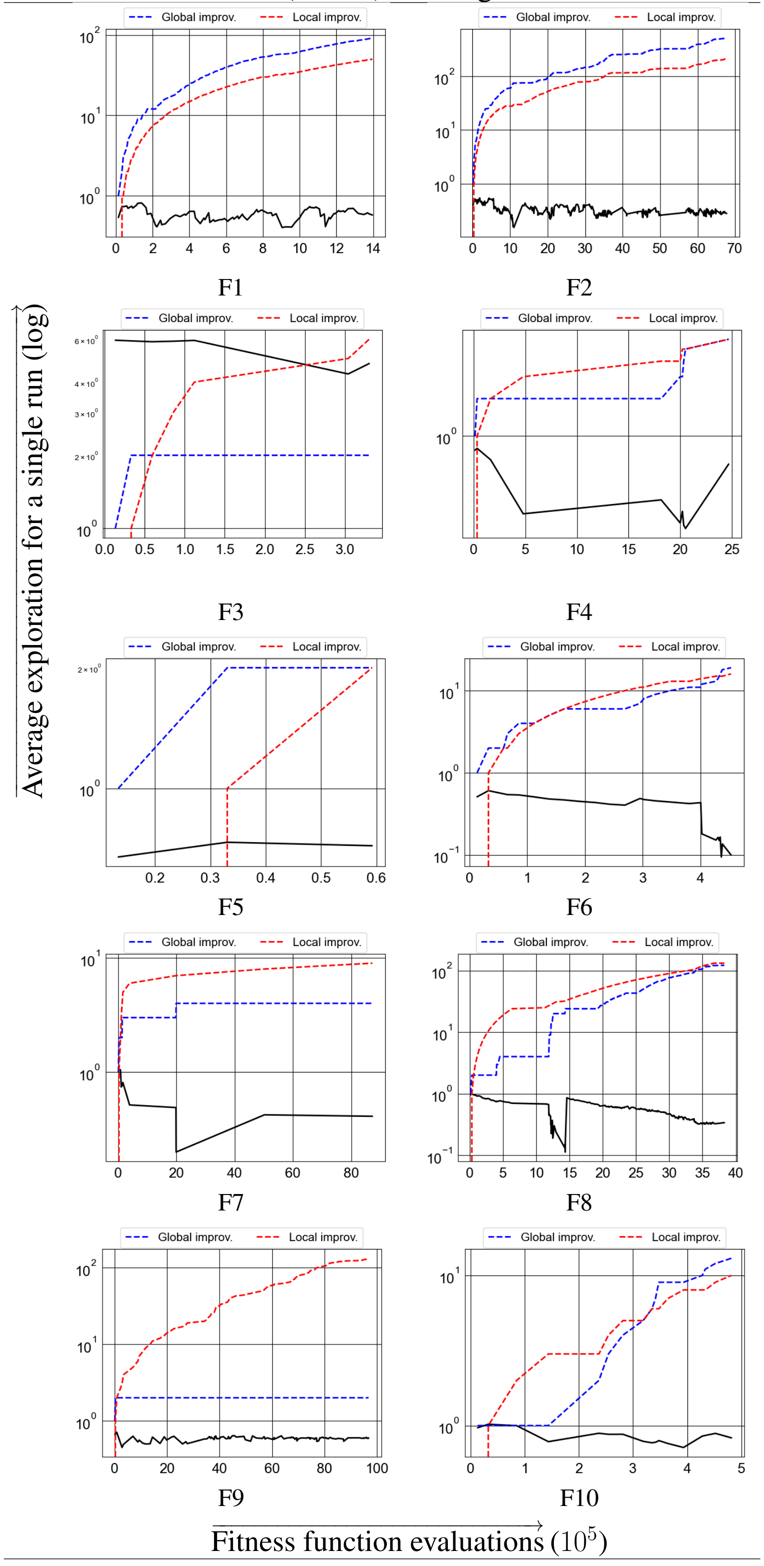
Source: From the author (2022).

of the tree throughout the optimization processes. Thus, comparing these plots with the previously presented in Figure 6.9, we observe that even though the MA has converged in all executions, it was also able to preserve a certain population diversity by noticing that the nodes have presented distinct best fitness values in different stages of the optimization. For instance, such scenarios are most evident in functions F3, F8, F9, and F10.

With this, to demonstrate the variability of the method in terms of Expr/Expt, Figure 6.11 shows the variation in the average exploration of all nodes of the tree, measured via the population diversity, throughout a single execution of the method for all benchmark functions (F1-F10). The exploration measure ($Expr\%$) was explained in Section 5.2.8 of the control procedure for convergence and performance. Thus, the exploration effort is calculated by Equation 5.5 and serves as the basis for the control of diversity and convergence. $Expr\%$ values are expressed as a percentage in the continuous range $[0, 1]$, where lower values of $Expr\%$ indicate low algorithm's exploration. We note that the exploitation effort is calculated as the complementary percentage to exploration, then addressing one of them is enough to deal with both efforts. Also, the figure illustrates the improvement process of the best solution over the evolutionary process. The dashed blue line in plots represents the growth of improvements made in the current best solution by global searches. On the contrary, the dashed red line in plots means the growth of improvements made in the best solution by local searches. The local searches are performed by the LIP of the MA (Section 5.2.6). In addition, Figure 6.12 illustrates the variation in the exploration values of each node of the tree throughout a single execution of the method for all benchmark functions (F1-F10). We observe that the y-axis in both figures is expressed in the logarithm scale.

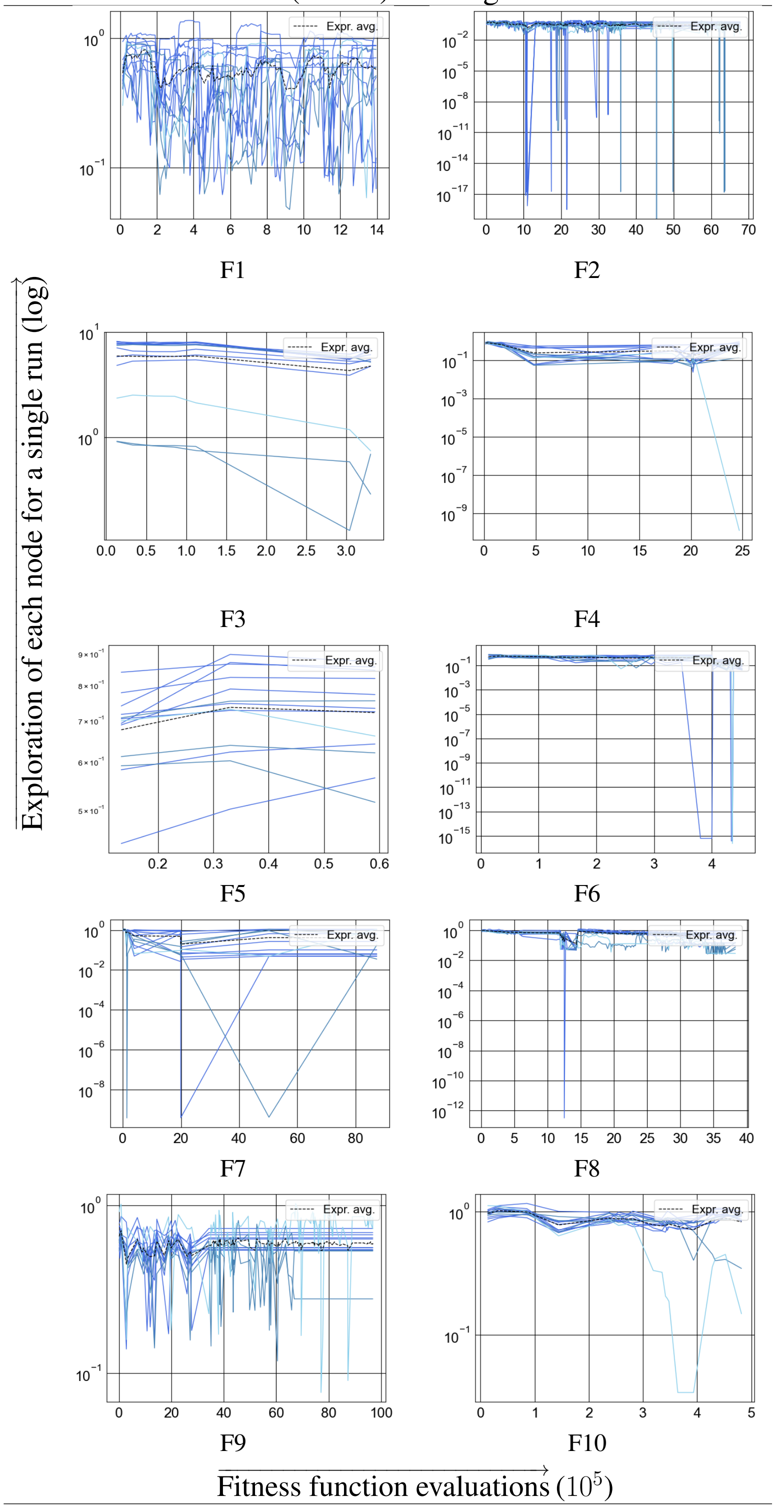
From the plots illustrated in Figures 6.11 and 6.12, it can be seen that the framework was able to regulate the Expr/Expt mechanisms throughout the optimization process. It has reached periods of higher exploration or exploitation at different stages of the execution, depending on a set of factors at each moment of the optimization, such as the population diversity degree, the increase/decrease in exploration, and its behavior over the generations given by the SMA, the number of improvements in solutions by global and local searches, and the number of nodes in the tree and their size. It is also possible to observe in Figure 6.12 the decrease in the number of tree nodes and the increase of the subpopulation size over time due to the TRP. We highlight that this mechanism increases the exploration of the remaining nodes but with more focused searches. Besides that, analyzing the plots in Figure 6.11, one can notice that the algorithm was capable of

Figure 6.11: Variation in the average exploration of all nodes of the tree (**y-axis**) throughout a single execution of the method (**x-axis**) for all global benchmark functions



Source: From the author (2022).

Figure 6.12: Variation in the exploration values of each node of the tree (**y-axis**) throughout a single execution of the method (**x-axis**) for all global benchmark functions



Source: From the author (2022).

continuously improving the best solution for both global and local searches.

Thereby, under the points of interest outlined for this thesis, it is possible to note that the algorithm was able to balance the Expr/Expt efforts while consistently improving the solutions. In general, through the search components implemented in the framework (Table 5.1) over a constructive design from intermediate variants (Table 6.3), such as the tree structure and niching strategies, inter and intra-niche optimization, TRP, LIP, control procedure for convergence and performance, DNSP, and hybrid-ABC, it managed to optimize the benchmark functions by exploring and refining the search space, as well as overcoming premature convergence to local optima in many scenarios.

Nonetheless, according to the results of Table 6.5, it is noticed that to enhance such results, the MA still needs improvements, mainly concerning the parameter control, i.e., an additional effort should be made to appropriately adapt the parameter setting to any function without the need to tune it for special cases. For instance, it is possible to include additional parameter control strategies not to have to tune the method for a specific benchmark function and better orchestrate all components of the framework, such as the number of layers, nodes, and individuals in the tree structure or the threshold used to regulate the Expr/Expt mechanisms.

Finally, in the next section, we describe the computational experiments concerning the benchmark functions of multimodal optimization. The MA framework was incremented with an external archive to preserve the optimal solutions found throughout the algorithm's execution.

6.3 Scenario of Multimodal Optimization

In this section, the MA framework with archive strategy, described in Section 5.3, was tested on benchmarks functions for multimodal optimization with more than one global optimum. This version was designed from the MA for global optimization discussed above. According to Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013), evolutionary metaheuristics in their original forms usually tend to converge to a single solution due to the adopted global components. However, most real problems are naturally multimodal with multiple satisfactory solutions. In such scenarios, the goal is to find as many global optimal or suboptimal solutions as possible so that a decision-maker can choose the most proper in his problem domain.

Thereby, to test our approach on this optimization case study, we used the test functions of the CEC 2013 for multimodal function optimization benchmark[2] [3] (LI; ENGELBRECHT; EPITROPAKIS, 2013). So the technical report of Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013) presents a benchmark suite composed of 20 multimodal functions with different characteristics for evaluating niching methods. It includes many identical functions with different dimension sizes.

The first ten functions are simple, well-known, and widely used functions. The remaining benchmark functions are more complex and follow the paradigm of composition functions defined in the CEC 2005 competition on single objective real-parameter optimization (SUGANTHAN et al., 2005). Composition functions are harder functions with several global optima. They are constructed as a weighted aggregation of n basic functions with different properties. Each one is shifted to a new position inside the optimization search space and can be either rotated through a linear transformation matrix or used as it is. The benchmark suite is detailed in Table 6.7.

Table 6.7: Summarization of the multimodal test functions

Function ID	**Function**	**Peak height**	**Nº global/local optima**	***D***	**Search range**	***r***
1	F1	200.0	2/3	1-D	[0, 30]	0.01
2	F2	1.0	5/0	1-D	[0, 1]	0.01
3	F3	1.0	1/4	1-D	[0, 1]	0.01
4	F4	200.0	4/0	2-D	[-6, 6]	0.01
5	F5	1.03163	2/2	2-D	[-1.9, 1.9]	0.5
6	F6	186.731	18/many	2-D	[-10, 10]	0.5
7	F7	1.0	36/0	2-D	[0.25, 10]	0.2
8	F6	2709.0935	81/many	3-D	[-10, 10]	0.5
9	F7	1.0	216/0	3-D	[0.25, 10]	0.2
10	F8	-2.0	12/0	2-D	[0, 1]	0.01
11	F9	0	6/many	2-D	[-5, 5]	0.01
12	F10	0	8/many	2-D	[-5, 5]	0.01
13	F11	0	6/many	2-D	[-5, 5]	0.01
14	F11	0	6/many	3-D	[-5, 5]	0.01
15	F12	0	8/many	3-D	[-5, 5]	0.01
16	F11	0	6/many	5-D	[-5, 5]	0.01
17	F12	0	8/many	5-D	[-5, 5]	0.01
18	F11	0	6/many	10-D	[-5, 5]	0.01
19	F12	0	8/many	10-D	[-5, 5]	0.01
20	F12	0	8/many	20-D	[-5, 5]	0.01

Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

According to Table 6.7, all test functions are formulated as twenty maximization problems. All functions are multimodal with dimensions (objective variables) varying from 1-D to 20-D, where the optimal objective variables are known for all benchmark

[2]<http://epitropakis.co.uk/cec20-niching/competition/>

[3]<http://epitropakis.co.uk/gecco2020/>

functions. The number of global optima in each function is shown in the fourth column, and their optimal values are indicated in the peak height column. The search range column defines the variable ranges for each function, where the same upper and lower bounds are adopted for all function dimensions. The last column represents the niche radius value that sufficiently distinguishes the two closest global optima for each function. The main characteristics of the benchmark problems are given by functions F1-F12, as follows (LI; ENGELBRECHT; EPITROPAKIS, 2013):

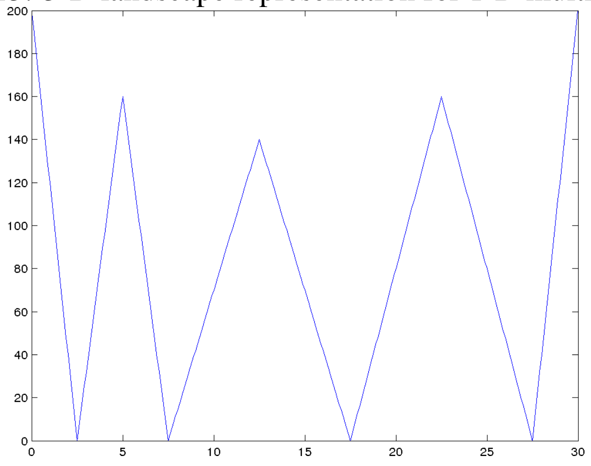
F1: Five-Uneven-Peak Trap Function (Equation 6.14):

$$F_1(x) = \begin{cases} 80(2.5 - x) & \text{for } (0 \leq x < 2.5), \\ 64(x - 2.5) & \text{for } (2.5 \leq x < 5.0), \\ 64(7.5 - x) & \text{for } (5.0 \leq x < 7.5), \\ 28(x - 7.5) & \text{for } (7.5 \leq x < 12.5), \\ 28(17.5 - x) & \text{for } (12.5 \leq x < 17.5), \\ 32(x - 17.5) & \text{for } (17.5 \leq x < 22.5), \\ 32(27.5 - x) & \text{for } (22.5 \leq x < 27.5), \\ 80(x - 27.5) & \text{for } (27.5 \leq x \leq 30) \end{cases} \tag{6.14}$$

Function properties:

- 1-D multimodal;
- Simple function;

Figure 6.13: 3-D landscape representation for 1-D multimodal F1



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

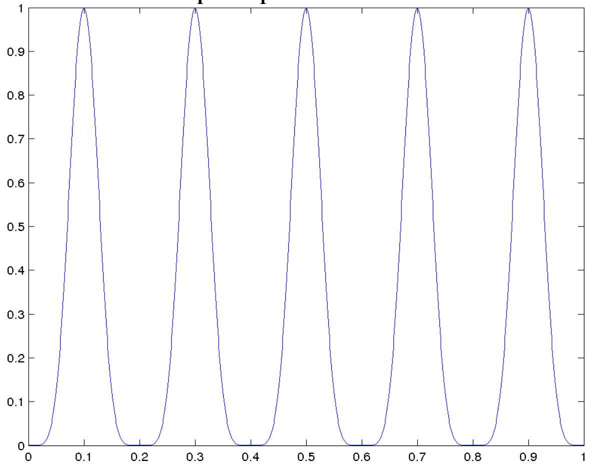
F2: Equal Maxima Function (Equation 6.15):

$$F_2(x) = \sin^6(5\pi x) \tag{6.15}$$

Function properties:

- 1-D multimodal;
- Simple function;

Figure 6.14: 3-D landscape representation for 1-D multimodal F2



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

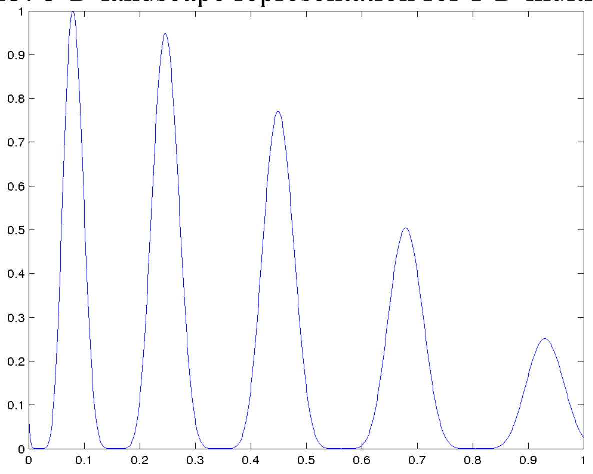
F3: Uneven Decreasing Maxima Function (Equation 6.16):

$$F_3(x) = \exp\left(-2\log(2)\left(\frac{x-0.08}{0.854}\right)^2\right)\sin^6\left(5\pi(x^{3/4}-0.05)\right) \tag{6.16}$$

Function properties:

- 1-D multimodal;
- Simple function;

Figure 6.15: 3-D landscape representation for 1-D multimodal F3



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

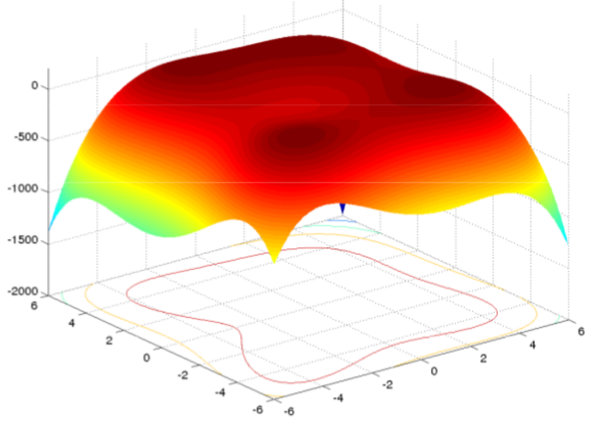
F4: Himmelblau Function (Equation 6.17):

$$F_4(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2 \tag{6.17}$$

Function properties:

- 2-D multimodal;
- Not scalable;

Figure 6.16: 3-D landscape representation for 2-D multimodal F4



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

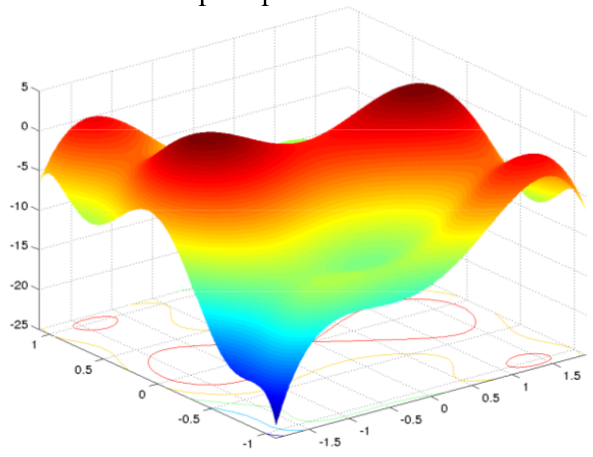
F5: Six-Hump Camel Back Function (Equation 6.18):

$$F_5(x,y) = -4\left[\left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (4y^2 - 4)y^2\right] \tag{6.18}$$

Function properties:

- 2-D multimodal;
- Not scalable;

Figure 6.17: 3-D landscape representation for 2-D multimodal F5



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

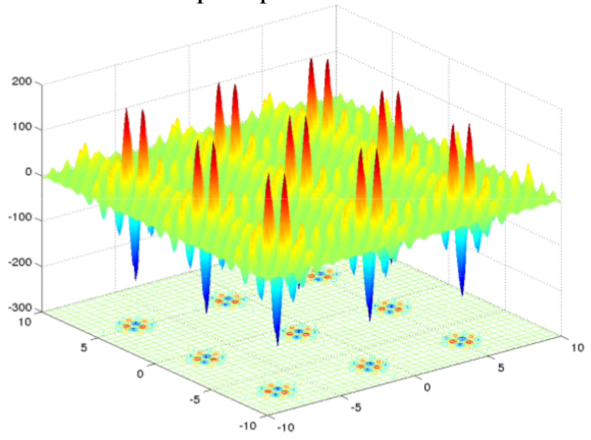
F6: Shubert Function (Equation 6.19):

$$F_6(x) = -\prod_{i=1}^{D}\sum_{j=1}^{5} j\cos\left((j+1)x_i + j\right) \tag{6.19}$$

Function properties:

- Multimodal;
- Scalable;
- Nº of global optima is given by the dimension D, such that $D \times 3^D$;

Figure 6.18: 3-D landscape representation for 2-D multimodal F6



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

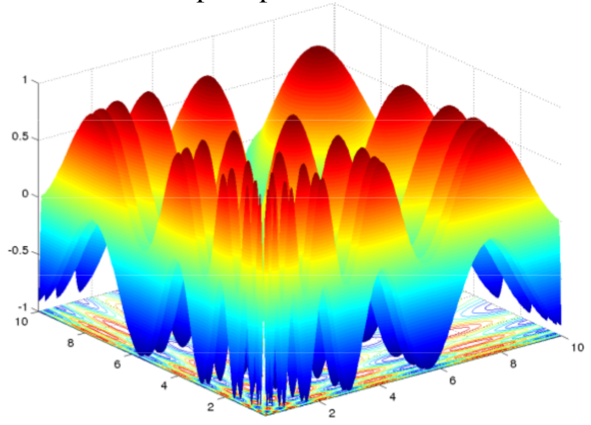
F7: Vincent Function (Equation 6.20):

$$F_7(x) = \frac{1}{D}\sum_{i=1}^{D} \sin\left(10\log(x_i)\right) \tag{6.20}$$

Function properties:

- Multimodal;
- Scalable;
- Nº of global optima is given by the dimension D, such that 6^D;

Figure 6.19: 3-D landscape representation for 2-D multimodal F7



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

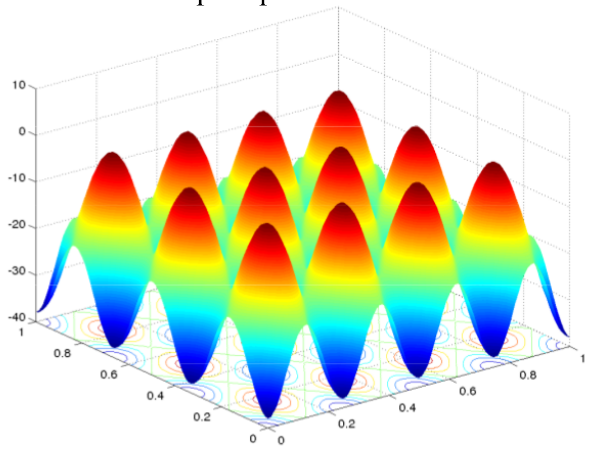
F8: Modified Rastrigin Function - All Global Optima (Equation 6.21):

$$F_8(x) = -\sum_{i=1}^{D} (10 + 9\cos(2\pi k_i x_i)) \tag{6.21}$$

Function properties:

- Multimodal;
- Scalable;
- Nº of global optima is defined by the benchmark, independent from D;

Figure 6.20: 3-D landscape representation for 2-D multimodal F8



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

Composition Functions: The following functions (F9-F12) are composition functions constructed as a weighted aggregation of basic functions. Thus, the pool of functions used to construct them includes five basic functions: Sphere, Griewank, Rastrigin, Weierstrass, and EF8F2 functions. These functions are described as follows.

S: Sphere function (Equation 6.22):

$$f_S(x) = \sum_{i=1}^{D} x_i^2 \tag{6.22}$$

G: Griewank's function (Equation 6.23):

$$f_G(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{6.23}$$

This function was also used in the previous scenario of single global optimization

(Section 6.2), described in function F5 by Equation 6.5.

R: Rastrigin's function (Equation 6.24):

$$f_R(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10) \tag{6.24}$$

This function was also used in the previous scenario of optimization (Section 6.2), described in function F4 by Equation 6.4.

W: Weierstrass function (Equation 6.25):

$$f_W(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{max}}\left[a^k\cos\left(2\pi b^k(x_i+0.5)\right)\right]\right) - D\sum_{k=0}^{k_{max}} a^k\cos(\pi b^k);$$
$$a = 0.5;\quad b = 3;\quad k_{max} = 20; \tag{6.25}$$

This function was also used in the previous scenario of optimization (Section 6.2), described in function F6 by Equation 6.6.

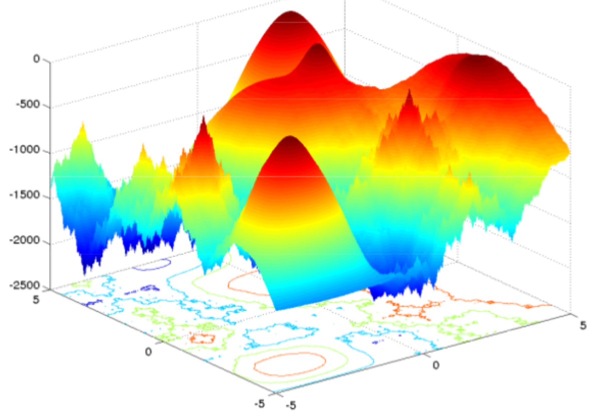
EF8F2: Expanded Griewank's plus Rosenbrock's function (Equation 6.26):

$$\begin{aligned} f_{EF8F2}(x) &= f_G f_{Rb}(x_1, x_2, \cdots, x_D); \\ f_G(x) &= \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1; \\ f_{Rb}(x) &= \sum_{i=1}^{D-1}(100(x^2 - x_{i+1})^2 + (x_i - 1)^2) \end{aligned} \tag{6.26}$$

F9: Composition Function 1 (CF1):

- Multimodal;
- Composed by Griewank, Weierstrass and Sphere basic functions;
- Shifted, non-rotated, non-symmetric, separable and scalable;
- Nº of global optima is defined by the benchmark, independent from D;

Figure 6.21: 3-D landscape representation for 2-D multimodal F9

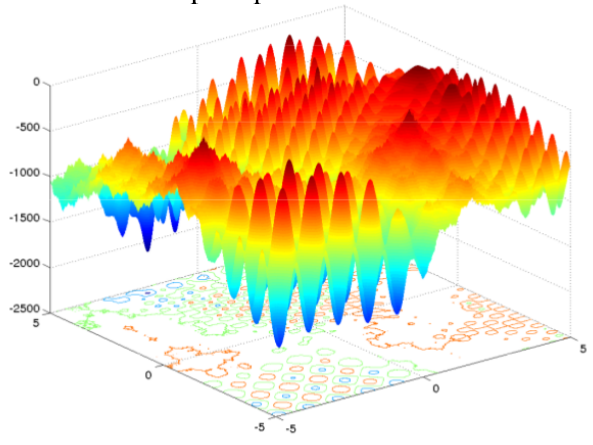


Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

F10: Composition Function 2 (CF2):

- Multimodal;
- Composed by Rastrigin, Griewank, Weierstrass and Sphere basic functions;
- Shifted, non-rotated, non-symmetric, separable and scalable;
- Nº of global optima is defined by the benchmark, independent from D;

Figure 6.22: 3-D landscape representation for 2-D multimodal F10

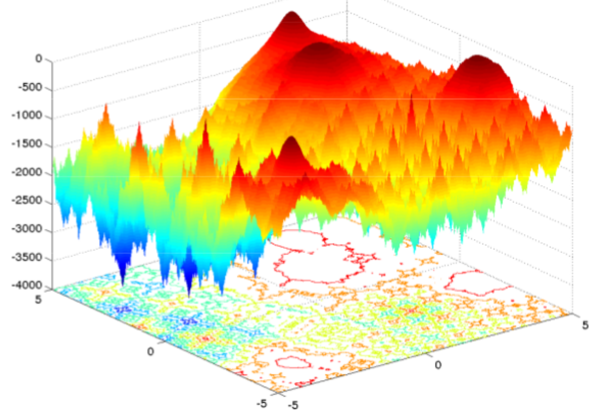


Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

F11: Composition Function 3 (CF3):

- Multimodal;
- Composed by EF8F2, Griewank and Weierstrass basic functions;
- Shifted, rotated, non-symmetric, non-separable and scalable;
- Nº of global optima is defined by the benchmark, independent from D;

Figure 6.23: 3-D landscape representation for 2-D F11

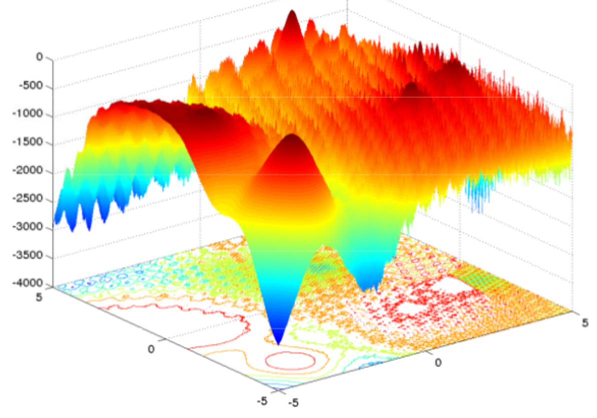


Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

F12: Composition Function 4 (CF4):

- Multimodal;
- Composed by Rastrigin, EF8F2, Griewank and Weierstrass basic functions;
- Shifted, rotated, non-symmetric, non-separable and scalable;
- Nº of global optima is defined by the benchmark, independent from D;

Figure 6.24: 3-D landscape representation for 2-D multimodal F12



Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

6.3.1 Evaluation Criteria

The multimodal benchmark suite is used to evaluate the ability of niching algorithms to locate global optima by using the functions shown in Table 6.7. It defines performance measures and procedures for comparing different methods (LI; ENGELBRECHT; EPITROPAKIS, 2013).

Thus, the report of Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013) provides a performance procedure to detect how many global optima have been found in a run for a given function. Firstly, the level of accuracy ($0 < \epsilon \leq 1$) is specified, where ϵ is a threshold value determining whether a global optimum has been found. Then, the information in Table 6.7 is used for each test function: the number of global optima, the fitness of the global optima (i.e., peak height), and the niche radius value that can sufficiently distinguish the two closest global optima. At the end of an optimization process, this procedure determines if a niching method has located all the global optima. Furthermore, since the exact number of global optima is known a priori, it is possible to measure a niching algorithm's performance in terms of the peak ratio and success rate for locating all global optima over multiple runs.

The peak ratio (PR) measure (THOMSEN, 2004) is used to evaluate the performance of a niching algorithm over multiple runs. Given a fixed maximum number of fitness function evaluations (Max_Evls) and a required accuracy level (ϵ), PR measures the average percentage of all known global optima found over multiple runs, as follows (LI; ENGELBRECHT; EPITROPAKIS, 2013):

$$PR = \frac{\sum_{i=1}^{N_{runs}} NPF_i}{N_{global} \cdot N_{runs}} \tag{6.27}$$

Where NPF_i denotes the number of global optima found at the end of the i-th execution, N_{global} is the number of known global optima, and N_{runs} is the number of runs. PR value equals one (1) means that all global optima have been found in all executions regarding the level of accuracy adopted.

In addition, the success rate (SR) measure is used to calculate the percentage of successful runs out of all runs. We note that a successful run is defined as a run where all known global optima are found.

$$SR = \frac{NSR}{N_{runs}} \tag{6.28}$$

Where NSR is the number of successful runs, SR value equals one (1) means that all executions were successful regarding the level of accuracy adopted.

The functions shown in Table 6.7 represent twenty maximization problems, where the goal is to find all existing global optima. In experimental settings, the report defines a fixed amount of Max_Evls as the stop criterion to evaluate the methods. The amount of Max_Evls used for each function is shown in Table 6.8. For each function, 50 consecutive runs of an algorithm are required, by adopting uniform random initialization within the given bounds. For PR (Equation 6.27) and SR (Equation 6.28) measures, it is defined five levels of accuracy (ϵ): $1.0E-01, 1.0E-02, 1.0E-03, 1.0E-04, 1.0E-05$. Also, the adopted niche radius are summarized in Table 6.7. Thus, in this work, the performance of the MA framework for each function is determined through the number of global optima found over the 50 runs and the PR and SR measures.

Table 6.8: Summarization of the maximum number of fitness function evaluations (Max_Evls) adopted for three ranges of multimodal benchmark functions

Range of functions	***Max_Evls***
F1 to F5 (1-D or 2-D)	$5.0E+04$
F6 to F11 (2-D)	$2.0E+05$
F6 to F12 (3-D or higher)	$4.0E+05$

Source: From Li et al. (LI; ENGELBRECHT; EPITROPAKIS, 2013).

6.3.2 Parameterization of the Method

Since its first version, the MA framework was designed for multimodal optimization, incorporating niching strategies and procedures for efficient Expr/Expt over the search space. Besides that, the MA, designed incrementally and tested in the previous section for global optimization, was complemented based on the issues concerned with discovering and maintaining the existing global optima of a given multimodal objective function. The method incorporated an external archive to store the identified global optima when converged regions of the search space are detected. The concept consists of storing the found optimal points and not losing them while locating new ones throughout the algorithm's execution. This version was detailed in Section 5.3.

This variant of the MA uses basically the same parameterization of the general framework for global optimization, which was detailed in Sections 5.2 and 6.2.1. Nonetheless, some of these parameters have been adjusted to better fit the method to the benchmark test functions for multimodal optimization and mainly to its limited computational budget (stop criterion) (Table 6.8). With this, the control parameters that were modified in comparison to the previously discussed parameterization for global optimization (Section 6.2.1) are described below:

- Number of layers in the tree structure: in this version, we adopted a tree structure with 5 layers ($N_{layers} = 5$), totaling the maximum of 121 nodes ($N_{nodes} = 121$). The number of layers and nodes were increased due to the complexity of the benchmark functions and the several existing global optima. It is noted that the goal of this optimization scenario is to find all global optima, then, a greater number of niches may increase the chance of locating them;
- Maximum number of individuals in the entire population of the MA: since this parameter also defines the maximum number of individuals in each node depending on the number of nodes in the tree, we adopted $Max\_NS = 1210$, and consequently the maximum of 10 individuals per node ($Max\_NS_{node} = 10$). As the number of nodes has been increased in an attempt to find more optimal solutions, the Max_NS_{node} was decreased to keep the algorithm's convergence and performance due to the limited budget for functions. This value was used to provide the method some flexibility when dynamic changing the niche size via the DNSP (Section 5.2.9), as the minimum number of individuals allowed in each node was kept in 5 ($Min\_NS_{node} = 5$);
- Number of generations executed by the core metaheuristic: this parameter represents the number of generations that the hybrid-ABC runs each generation of the framework. g_{core} was defined as 1 due to the limited computational budget defined in the benchmark suite and the higher number of nodes adopted. This value is used to allow inter and intra niche exploration without compromising any of them;
- Number of generations used to calculate the SMA: this parameter defines the period in terms of generations used to calculate the SMA in the control procedure for convergence and performance (Section 5.2.8). We adopted $n\_gen = 14$ to consider the exploration degree over a certain number of generations, since $n\_gen = 14$ also implies in 14 generations of the core metaheuristic;
- In this version, the linear reduction strategy of the number of tree nodes via the TRP (Section 5.2.7) was disabled in an attempt to locate and maintain as many global optima as possible throughout the run;
- In the LIP (Section 5.2.6), the constant used to regulate the LS intensity (I_{str}) every time that the SW algorithm is performed was set as $I_{str} = 300$ fitness function evaluations instead of $I_{str} = 1000$ as in the previous version. The parameter was

changed to better balance the Expr/Expt efforts on global and local searches due to the limited budget for each run;

- In the niche restarting (Section 5.2.8) and LS restarting (Section 5.2.6) procedures, the algorithm stores the entire subpopulation of a node into the external archive if it has stagnated, but now including the gbest solutions, and generates a new one;
- Stop condition: the stop condition is defined by the number of fitness function evaluations performed throughout an execution. Table 6.8 defines the Max_Evls for each test function.

6.3.3 Results and Discussion

As mentioned above, all tests in this section were executed 50 times for each benchmark function. The obtained results of the MA framework are detailed in Table 6.9 for the five levels of accuracy adopted. The table shows the PR and SR values obtained for the corresponding accuracy.

Analyzing the results of Table 6.9, we observe that the MA was able to find all global optima for 14 out of 20 functions with $\epsilon = 1.0E-01$, which is indicated by the PR and SR values equal to 1. Also, the method has reached PR values higher than 90% for 18 functions with $\epsilon = 1.0E-01$, which may indicate reasonable exploration ability to locate and preserve optimal solutions. One reason is probably the niching components included in the framework, the modifications in control parameters to fit the method to the benchmark test functions for multimodal optimization, and the external archive used to store the converged solutions during the execution.

Considering the obtained results for the highest level of accuracy ($\epsilon = 1.0E-05$) in Table 6.9, the algorithm has found all global optima for 7 out of 20 problems regarding the PR and SR values. It was able to reach PR values higher than 90% for 9 functions with $\epsilon = 1.0E-05$. It is noted that the higher the level of accuracy, the more refined the solution must be over the optimization process. Thus, such results may indicate that for the hardest cases, the algorithm was able to find the global optima's neighborhood but could not properly exploit these found optima to achieve more accurate solutions. For instance, the algorithm has reached PR values equal to 1 with $\epsilon = 1.0E-01$ for some problems. However, with the increase in accuracy, the PR has decreased. Besides that, we highlight test case 20 of F12 (20-D), where the method has achieved the lowest PR values

Table 6.9: Summarization of the results regarding peak ratios and success rates considering 50 runs of the MA framework for all test functions

Acc. level ϵ	**1 - F1 (1-D)**		**2 - F2 (1-D)**		**3 - F3 (1-D)**		**4 - F4 (2-D)**		**5 - F5 (2-D)**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
`1.0E-01`	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
`1.0E-02`	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
`1.0E-03`	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
`1.0E-04`	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
`1.0E-05`	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
Acc. level ϵ	**6 - F6 (2-D)**		**7 - F7 (2-D)**		**8 - F6 (3-D)**		**9 - F7 (3-D)**		**10 - F8 (2-D)**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
`1.0E-01`	1.000	1.000	1.000	1.000	0.992	0.660	0.624	0.000	1.000	1.000
`1.0E-02`	1.000	1.000	0.462	0.000	0.991	0.660	0.219	0.000	1.000	1.000
`1.0E-03`	1.000	1.000	0.245	0.000	0.991	0.640	0.120	0.000	1.000	1.000
`1.0E-04`	1.000	1.000	0.245	0.000	0.990	0.640	0.119	0.000	1.000	1.000
`1.0E-05`	1.000	1.000	0.245	0.000	0.989	0.600	0.119	0.000	1.000	1.000
Acc. level ϵ	**11 - F9 (2-D)**		**12 - F10 (2-D)**		**13 - F11 (2-D)**		**14 - F11 (3-D)**		**15 - F12 (3-D)**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
`1.0E-01`	1.000	1.000	0.990	0.920	0.990	0.940	1.000	1.000	1.000	1.000
`1.0E-02`	0.873	0.320	0.988	0.900	0.673	0.000	0.667	0.000	0.728	0.000
`1.0E-03`	0.790	0.120	0.988	0.900	0.667	0.000	0.667	0.000	0.728	0.000
`1.0E-04`	0.780	0.120	0.988	0.900	0.667	0.000	0.667	0.000	0.728	0.000
`1.0E-05`	0.773	0.100	0.985	0.880	0.667	0.000	0.667	0.000	0.728	0.000
Acc. level ϵ	**16 - F11 (5-D)**		**17 - F12 (5-D)**		**18 - F11 (10-D)**		**19 - F12 (10-D)**		**20 - F12 (20-D)**	
	PR	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**	**PR**	**SR**
`1.0E-01`	1.000	1.000	1.000	1.000	1.000	1.000	0.828	0.440	0.102	0.000
`1.0E-02`	0.667	0.000	0.700	0.000	0.667	0.000	0.480	0.000	0.058	0.000
`1.0E-03`	0.667	0.000	0.700	0.000	0.667	0.000	0.362	0.000	0.035	0.000
`1.0E-04`	0.667	0.000	0.688	0.000	0.667	0.000	0.345	0.000	0.007	0.000
`1.0E-05`	0.667	0.000	0.675	0.000	0.667	0.000	0.310	0.000	0.003	0.000

Source: From the author (2022).

among all other functions. Problem 20 presents the highest dimensionality and could indicate a method's drawback in dealing with high-dimensional functions. Moreover, in cases 15, 17, and 19, which also consist of the F12 with lower dimensions, the method has obtained better results, locating all global optima with $\epsilon = 1.0E - 01$.

Therefore, to situate our methods according to relevant methods in this scenario, we compared the MA framework with three other niching methods using the multimodal benchmark functions. We adopted the PR values with a level of accuracy equal to $1.0E - 05$ as a key criterion to rank algorithms, as shown in the benchmark's report (LI; ENGELBRECHT; EPITROPAKIS, 2013). The first method is the well-known Crowding DE (CDE) (THOMSEN, 2004), which was presented as a baseline model in the technical report (LI; ENGELBRECHT; EPITROPAKIS, 2013). CDE is a niching algorithm that uses a crowding strategy to enforce the population diversity and, therefore, to prevent premature convergence to an optimum. The second method is the RS-

CMSA (AHRARI; DEB; PREUSS, 2017b; AHRARI; DEB; PREUSS, 2017a), which is a niching algorithm that runs many subpopulations in parallel through instances of the core search algorithm CMSA, such that solutions maintain a distance from a number of taboo points which are stored in an internal archive and regulated by the HV test. The RS-CMSA was evaluated on these benchmark functions for multimodal optimization (LI; ENGELBRECHT; EPITROPAKIS, 2013), where it was the winner of the GECCO 2017 competition on niching methods for multimodal function optimization[4]. The third method is the HillVallEA-AMu, which uses the HV Clustering algorithm to adaptively cluster the search space in niches such that a single optimum resides in each niche. It runs the core search algorithm AMaLGaM-Univariate (BOSMAN; GRAHL; THIERENS, 2013) with a restart scheme. HillVallEA-AMu was the winner of the GECCO 2018/2019 competitions on niching methods for multimodal function optimization[5] [6]. We observe that the three algorithms were previously detailed in Section 4.6.2 of relevant niching algorithms.

Thus, Table 6.10 shows the PR values with $\epsilon = 1.0E-05$ of each method for all test functions. The results of Table 6.10 were extracted from the corresponding published works. Figure 6.25 illustrates the bar charts related to the PR values of Table 6.10 for the four compared methods, where M1 (gray) represents the proposed MA framework, M2 (green) is the CDE, M3 (red) is the RS-CMSA, and M4 (blue) represents the HillVallEA-AMu.

Regarding the results of Table 6.10, it can be seen that RS-CMSA was the best performing algorithm with an average peak ratio of 0.856, followed by HillVallEA-AMu with an average PR of 0.847 over all test functions. Our proposed method obtained an average PR of 0.725. The worst performing algorithm was the CDE, with an average PR of 0.435. The MA outperformed CDE in almost all cases, but it still needs improvements in certain aspects, such as exploitation of solutions and dimensionality of variables, to approach the top algorithms.

Thereby, Figure 6.26 shows the method's convergence in the function of the best fitness values of each node of the tree for all benchmark functions (1-20) throughout the 50 runs. We note that each color in the plots represents the nodes of each layer of the tree, and each execution of the method could use the maximum of 121 nodes ($N_{nodes} = 121$) without the TRP procedure.

From the convergence plots in Figure 6.26, it is observed that the MA was able to

[4]<http://www.epitropakis.co.uk/gecco2017/>
[5]<http://www.epitropakis.co.uk/gecco2018/>
[6]<http://www.epitropakis.co.uk/gecco2019/>

Table 6.10: Summarization of the peak ratio values with $\epsilon = 1.0E-05$ for the MA framework, CDE, RS-CMSA, and HillVallEA-AMu considering 50 runs for all test functions. The average peak ratio is computed for all benchmark function

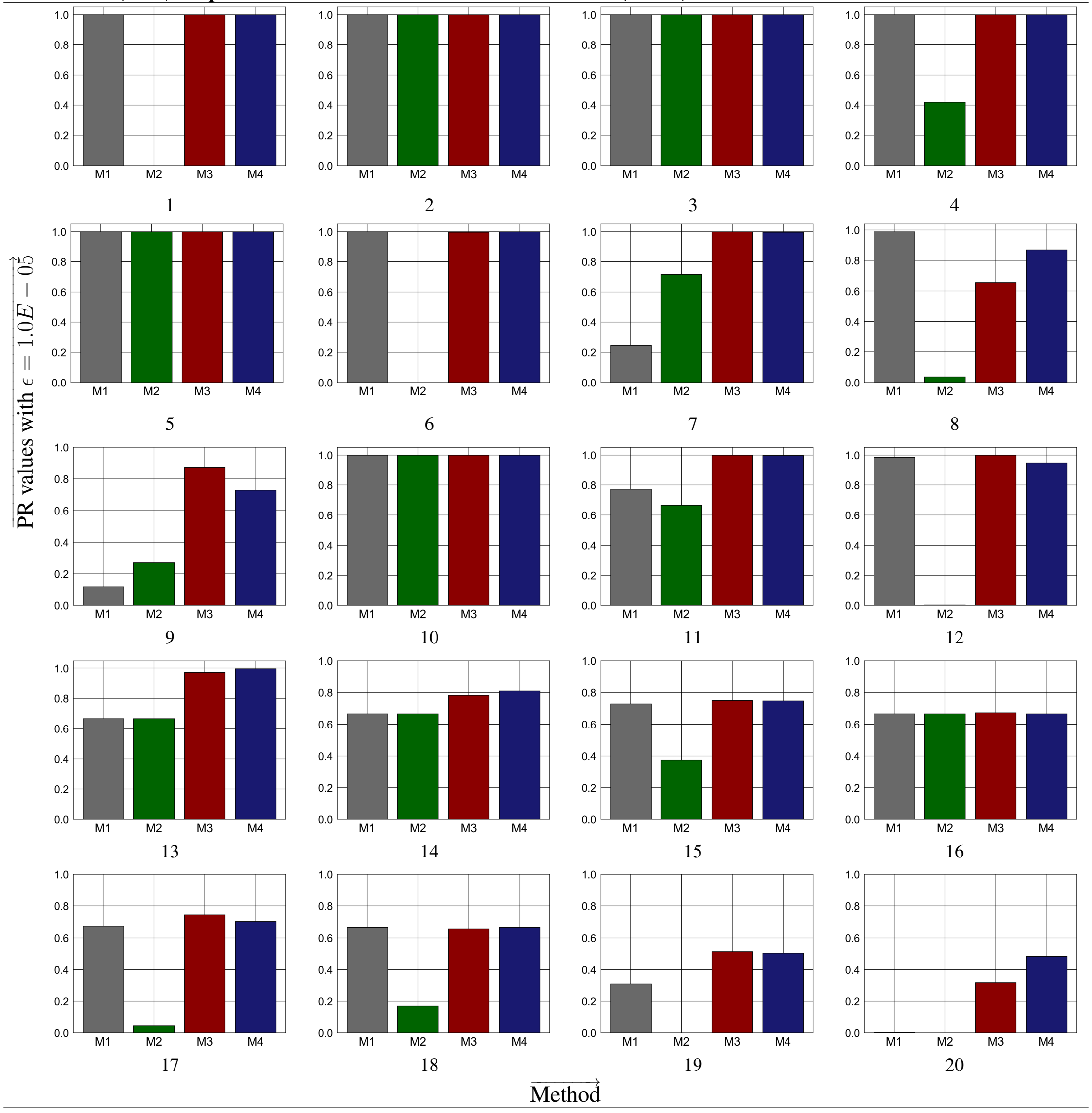
Function ID	**Method**			
	MA	**CDE**	**RS-CMSA**	**HillVallEA-AMu**
1	**1.000**	0.000	**1.000**	**1.000**
2	**1.000**	**1.000**	**1.000**	**1.000**
3	**1.000**	**1.000**	**1.000**	**1.000**
4	**1.000**	0.420	**1.000**	**1.000**
5	**1.000**	**1.000**	**1.000**	**1.000**
6	**1.000**	0.000	0.999	0.997
7	0.245	0.716	0.997	**1.000**
8	**0.989**	0.038	0.871	0.656
9	0.119	0.270	0.730	**0.875**
10	**1.000**	**1.000**	**1.000**	**1.000**
11	0.773	0.667	0.997	**1.000**
12	0.985	0.002	0.948	**1.000**
13	0.667	0.667	**0.997**	0.973
14	0.667	0.667	**0.810**	0.783
15	0.728	0.375	0.748	**0.750**
16	0.667	0.667	0.667	**0.673**
17	0.675	0.047	0.703	**0.745**
18	**0.667**	0.170	**0.667**	0.657
19	0.310	0.000	0.503	**0.512**
20	0.003	0.000	**0.483**	0.318
Average	0.725	0.435	0.856	0.847

Source: From the author (2022).

manage the Expr/Expt efforts according to the optimization processes and has converged to optimal regions while keeping the population diversity in an attempt to find more distinct global optima. The plots show the high variability of the tree nodes in terms of best fitness values, which demonstrates that the subpopulations have reached periods of higher diversity or convergence at different stages of the method's execution. It is known that such balance between Expr/Expt is required and expected for multimodal optimization as its goal is to find all existing global optima. Nonetheless, we note that the plots only show the partial convergence of the algorithm since all global optima of the same function have the same peak height, as shown in Table 6.7. With this, the number of global optima found by the method for each function and its performance were previously discussed from the PR values shown in Table 6.9.

Therefore, corroborating with the previous analysis from the obtained PR values

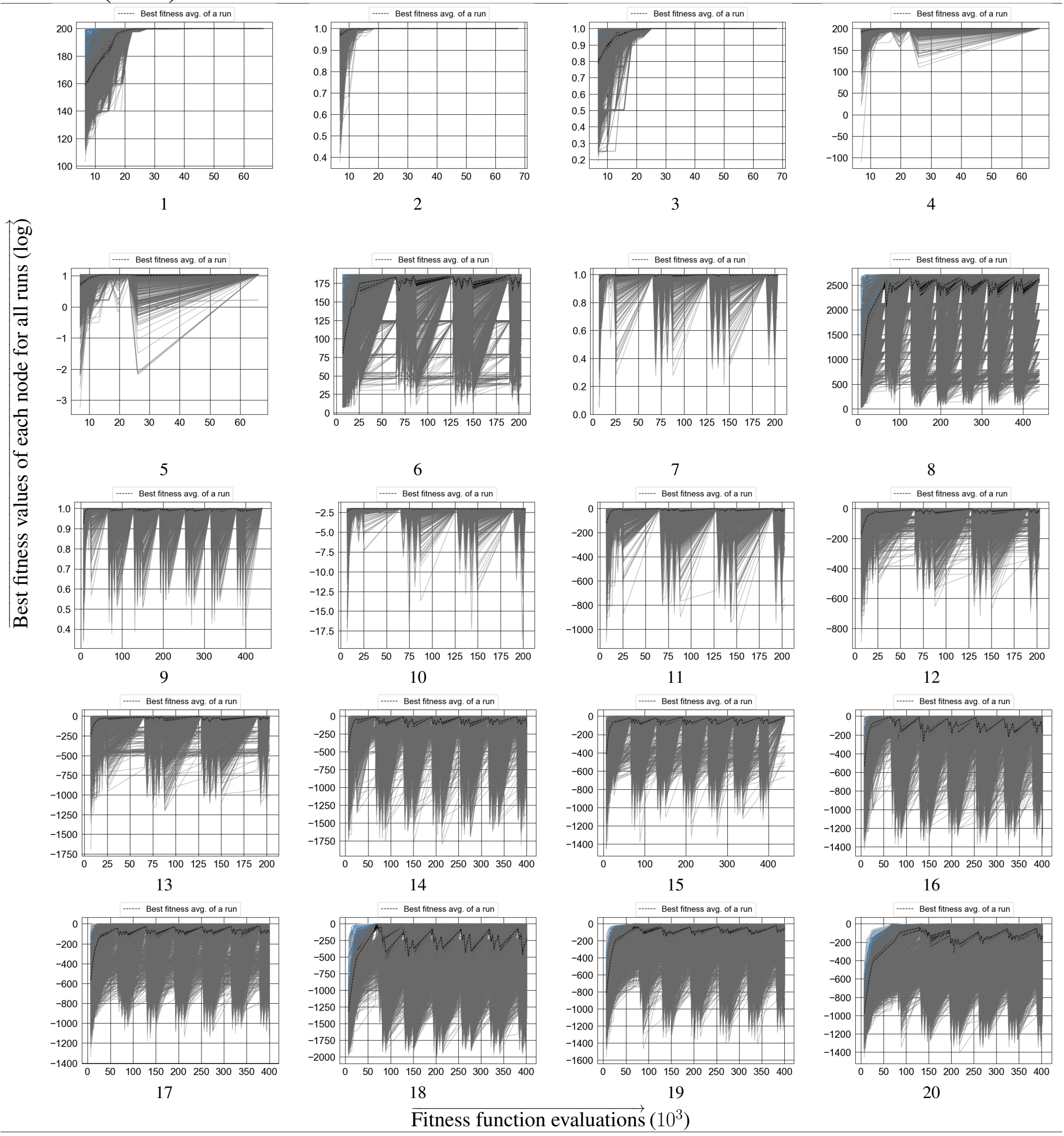
Figure 6.25: Peak ratio values with $\epsilon = 1.0E-05$ for the four compared methods over 50 runs for all test functions, where M1 (gray) is the MA framework, M2 (green) is the CDE, M3 (red) represents the RS-CMSA and M4 (blue) is the HillVallEA-AMu



Source: From the author (2022).

(Tables 6.9 and 6.10), the convergence plots indicate the algorithm's exploratory ability to discover and maintain optimal solutions while locating new ones throughout the execution process, which is one of the main issues of multimodal optimization (LI et al., 2016; WANG et al., 2019). However, from the same results, one can notice that our method could not properly refine the found global optima in some cases when compared with the top algorithms. For instance, it can be seen from the plots of the hardest functions that the method has kept a higher population diversity degree over the entire optimization. We observe the variation among the best fitness values of the nodes, which may demonstrate that optimal solutions have not been fully refined. Based on this, to overcome such an

Figure 6.26: Convergence of the MA regarding the best fitness values of each node of the tree (**y-axis**) throughout each of the 50 runs (**x-axis**) for the multimodal benchmark functions (1-20)



Source: From the author (2022).

issue in future works, one could enhance the exploitation ability of the MA by focusing more on LS by regulating the LIP procedure (Section 5.2.6) for this purpose, such as increasing the LS intensity control parameter (I_{str}) or better selecting the individuals for LS.

Another drawback is that the method could not perform well in problem 20, with the highest dimensionality in the benchmark. This result suggests that components for high-dimensional optimization could be considered to overcome the multimodality issues of larger search spaces. In addition, another improvement to MA for overall performance could consist of using a more robust archive strategy to deal with the found optimal solutions. For instance, we could consider some archived solutions in the search operations as positive feedback to the evolutionary process, as shown in AM-ACO (YANG et al., 2016b) and AGDE (ZHAO; ZHAN; ZHANG, 2020). Besides, it is possible to perform a refinement process in the archive to increase the solutions' accuracy as in DIDE (CHEN et al., 2019) or adopt archived taboo points combined with the HV test to detect solutions in the same optimal region as in RS-CMSA and HillVallEA. One could also test a different number of layers in the tree since a more robust archive strategy may enhance the method's abilities of Expr/Expt.

Finally, the MA framework performed well on the CEC 2013 benchmark for multimodal function optimization in general, except for the aforementioned problems. Thus, in the next section, we detail the computational experiments for the final optimization scenario, which is the 3-D PSP problem.

6.4 Prediction of 3-D Protein Structures

This section presents the obtained results concerning the performance of the proposed MA framework for the 3-D PSP problem, which was defined as our real case study for multimodal continuous optimization. As discussed earlier, the prediction of 3-D protein structures represents one of the key problems in Structural Bioinformatics, since the protein's function is directly related to its assumed conformation (DILL; MACCALLUM, 2012). Protein folding can provide to researchers valuable understanding of the protein roles in the cell (BRANDEN; TOOZE, 1999; LASKOWSKI; WATSON; THORNTON, 2005). Nonetheless, the PSP is classified as NP-hard problem (UNGER; MOULT, 1993; CRESCENZI et al., 1998) due to the high dimensionality of variables and its multimodal search space. Hence, structure modeling as computational optimization can be seen as a way to overcome some of the PSP complexities and ease the protein structure-based studies.

Thereby, the MA framework was incremented based on a constructive perspective from the versions presented in Sections 5.2 and 5.3 for global and multimodal optimization, respectively. This version included the stage of sampling models and initialization of solutions, previously described in Section 5.4.2, integrated into the MA framework, which yields the optimization stage of the structures coming from the sampling step, discussed in Section 5.4.3. The first stage was created to initially explore the conformational search space considering some insights about previously known protein structures, aiming to locate different structural groups for target proteins. In the second stage, the algorithm uses as initial individuals the solutions from the first step, aiming to deal with diversified and reasonable solutions since the beginning of the method's evolutionary process. Besides the components implemented to deal with the multimodal objective function, the framework was designed to consider the problem-specificities and their optimization challenges. It incorporated problem-dependencies and data knowledge from experimentally determined 3-D protein structures to turn it more robust to the problem under study (Section 5.4).

With this, for the prediction process of 3-D protein structures, the MA was tested on the set of target proteins described in Table 6.11, which comprises 15 distinct sequences ranging from 29 to 91 amino acids. These targets were selected to ensure that the test set encompasses amino acid sequences of different sizes and distinct topologies. The protein sequences were obtained from the PDB and used as case studies to test the proposed algorithm. Table 6.11 shows information about each target protein, such as size and SS content. We note that the target protein T0820 was extracted from the test set presented in the CASP11[7] experiments, which was referenced as T0820-D1 and categorized in the FM category (KINCH et al., 2016).

6.4.1 Evaluation Criteria and Parameterization of the Method

This optimization scenario aims to evaluate the method's potential in facing a complex real-world multimodal problem and consolidate the results obtained in the previous scenarios. The experiments were also conducted to analyze the method's behavior regarding the biological significance (quality) of the best solutions found, as well as to situate it in comparison to one of the state-of-the-art algorithms for PSP, the method of Rosetta (ROHL et al., 2004; SONG et al., 2013), described in Section 4.7.1. The Rosetta was chosen according to the latest CASP reports (MOULT et al., 2018; ABRIATA et al., 2018), which have pointed it out as one of the most relevant methods in the field, regarding automatic techniques of FM prediction without manual intervention (*servers*), due to the achieved results in the CASP competitions over the years. Nevertheless, it is observed that currently, the state-of-the-art method for PSP is the AlphaFold from DeepMind (SENIOR et al., 2019; SENIOR et al., 2020; JUMPER et al., 2020) (Section 4.7.2). However,

[7]<predictioncenter.org/casp11/targetlist.cgi>

Table 6.11: Set of target proteins used in the computational experiments to predict the 3-D structure of proteins. The second column shows the number of amino acid residues, and the third shows the secondary structure components

PDB ID	**Target Length**	**SS content**
1AB1	46	1 β-sheet/2 helices
1ACW	29	1 β-sheet/1 helix
1AIL	70	3 helices
1DFN	30	1 triple β-sheet
1ENH	54	3 helices
1FNA	91	2 β-sheets (1 quadruple)
1OPD	85	1 quadruple β-sheet/3 helices
1Q2K	31	1 β-sheet/1 helix
1ROP	56	2 helices
1UTG	70	5 helices
2MR9	44	3 helices
2P5K	64	1 β-sheet/3 helices
2PMR	76	3 helices
3V1A	48	2 helices
T0820 (CASP11)	90	3 helices

Source: From the author (2022).

the algorithm is not yet widely available to the scientific community.

In experimental settings, both the MA framework and Rosetta algorithms were run 8 times for each target of the benchmark proteins (Table 6.11). We adopted as stop criterion for our method the maximum of 10^6 fitness function evaluations ($Max\_Evls = 1.0E + 06$) per run on each target. The Rosetta[8] was run from the available software version (Rosetta commons, Academic License, Version 3.4), and the recommended default settings have been used (ROHL et al., 2004). We present a structural analysis of the solutions for each case study among the 8 runs performed. The structural quality of the predicted structures was carried out by similarity comparisons with experimentally determined protein structures regarding the RMSD (ZHANG; SKOLNICK, 2004) and the Global Distance Total Score (GDT_TS) test (ZEMLA, 2003). Such structural measures were used to evaluate the performance of the methods to predict 3-D protein structures.

The RMSD is widely used to assess the similarity degree between two 3-D protein structures and characterizes a minimization function (i.e., RMSD value equal to 0 indicates identical structures). This measure can be expressed according to Equation 6.29. The RMSD computation only considers the protein backbone C_α atoms of the 3-D struc-

[8]<www.rosettacommons.org>

tures under comparison.

$$RMSD(a,b) = \sqrt{\frac{\sum_{i=1}^{n} \|r_{ai} - r_{bi}\|^2}{n}} \tag{6.29}$$

Where a and b represent the two structures under comparison, n represents the amino acid sequence size, r_{ai} and r_{bi} are vectors describing the Cartesian positions of the atom i in the structures a and b, respectively. It is considered in the RMSD calculation that a and b were previously superimposed optimally.

In addition, the GDT_TS evaluates the structural similarity between two 3-D protein structures, such as the RMSD. We note that, unlike the RMSD, the GDT_TS is a maximization measure (i.e., GDT_TS value equal to 100 indicates identical structures). The GDT_TS can be expressed by Equation 6.30.

$$GDT_{TS} = \frac{(GDT_{P1} + GDT_{P2} + GDT_{P4} + GDT_{P8})}{4} \tag{6.30}$$

Where GDT_{Pn} represents the percentage of amino acids under the distance threshold $\leq n$.

Parameter setting: In this work, the PSP was computationally modeled as a global optimization problem with a multimodal objective function. Thus, the best solution for each algorithm run is the one with the lowest fitness value at the end of the optimization process among all subpopulations. We adopted the same scheme to determine the best solution for the Rosetta. Results were compared by the RMSD and GDT_TS measures regarding the experimentally determined protein structures.

To estimate the folding condition of a solution throughout the optimization process, we adopted as fitness function the composite energy function (E_{final}) described by Equation 3.12 in Section 3.7. This function represents a minimization problem and aims to differentiate between more or less stable protein models. Theoretically, conformations around a native state must reflect global minimal regions of their free energy (ANFINSEN, 1973). The multimodal function E_{final} is characterized by the highly rough search space with several valleys seen as local and global optima (HANDL; LOVELL; KNOWLES, 2008). It comprises the summation of three distinct functions: the Rosetta energy function (ROHL et al., 2004) (Section 3.7.1), the CM (Section 3.7.2) and SS (Section 3.7.3) functions.

Therefore, to test the proposed framework for the prediction of 3-D protein structures, we used the same parameter setting presented in the scenario of global optimization (Section 6.2) since our primary goal is to find the model with the lowest fitness value, which tends to represent the best solution. However, the second scenario of multimodal optimization (Section 6.3) was used to highlight the strengths and weak points of the method to be addressed in future works, such as the lack of exploitation and the dimensionality problem concerning harder multimodal functions with more than one global optimum. Thus, as the obtained results from the previously optimization scenarios have demonstrated, with exceptions, satisfactory results when compared with the state-of-the-art algorithms, the framework preserved the exact mechanisms previously described for global optimization combined with the specific-problem strategies discussed in Section 5.4.3. Such knowledge-based strategies are fundamental to support the method over the search process and overcome, to a certain degree, the previously pointed out shortcomings concerning global and multimodal optimization. This MA version receives as additional input parameters the primary and secondary structures of the target protein. The only modified parameter was the stop criterion of the algorithm, which was set as $Max\_Evls = 1.0E + 06$ per run on each target.

6.4.2 Results and Discussion

This section presents and discusses the obtained results of the proposed MA framework applied to the PSP problem. The results were evaluated via the RMSD and GDT_TS measures regarding the experimentally determined protein structures and also compared with the models predicted by the method of Rosetta. Table 6.12 summarizes the RMSD and GDT_TS results obtained for the 8 runs of the MA and Rosetta for the set of 15 target proteins described in Table 6.11. The best results for each target protein reached by MA and Rosetta methods were compared using the non-parametric *Mann-Whitney* similarity test (U-test) for independent samples. We adopted the significance level of 5%. The p-value resulting from the test determines the similarity between two samples. Values above 5% reject the similarity hypothesis, and the samples are considered statistically different. The significantly different samples are highlighted with “+” in the table. The **boldface** numbers represent the best results regarding RMSD and GDT_TS for each target.

Regarding the obtained results from Table 6.12, it can be observed by the highlighted numbers that our method has reached better results than Rosetta for many target proteins. It was also able to predict protein structures with similar folding to the experimentally determined ones. It is highlighted that predicted structures can be considered

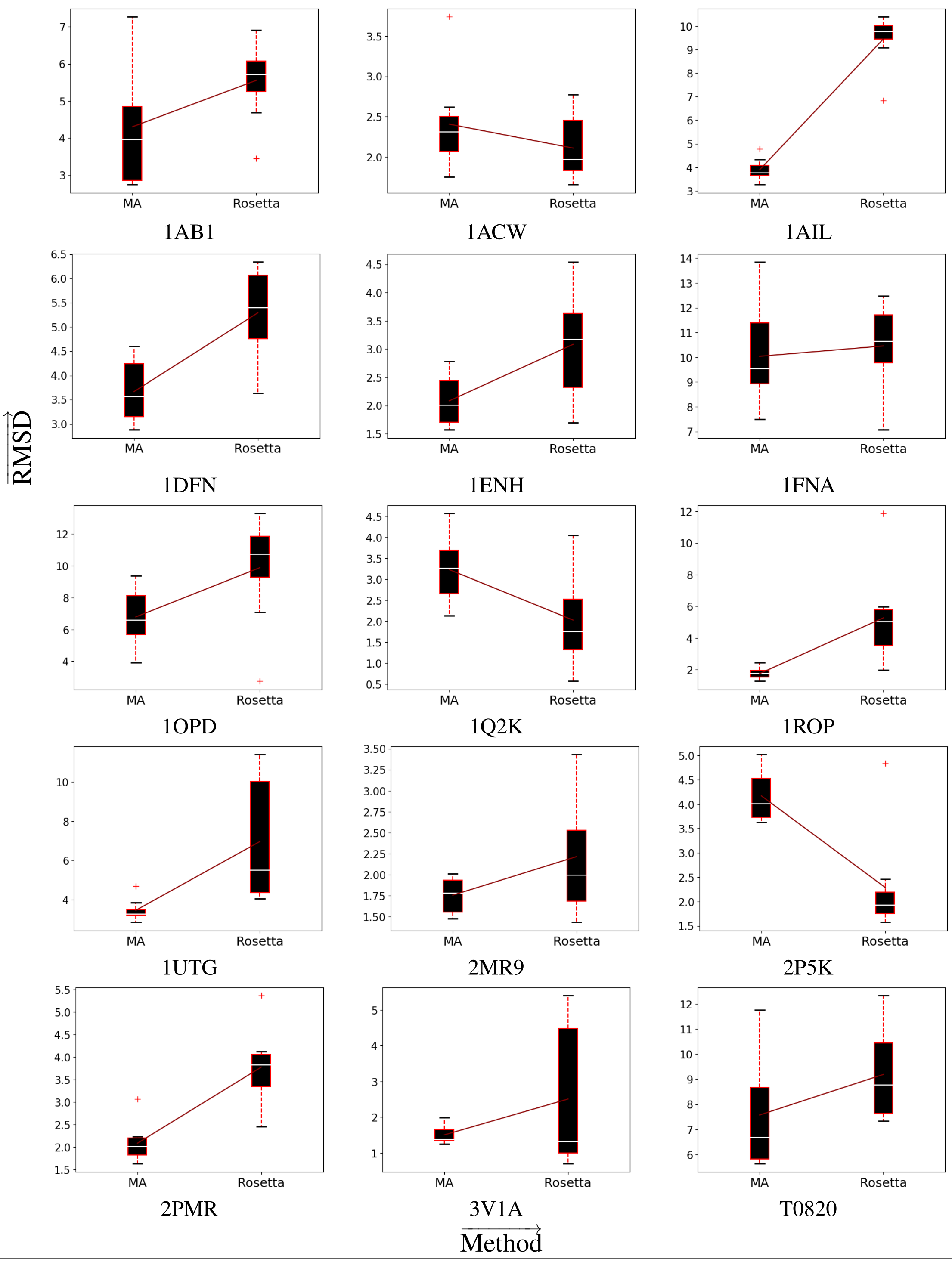
Table 6.12: Summary of the results obtained regarding the RMSD and GDT_TS for the 8 runs of the MA and Rosetta. The **boldface** numbers are the best results regarding RMSD and GDT_TS. The p-value column represents the result of p from the non-parametric *Mann-Whitney* test performed between MA and Rosetta concerning the RMSD and GDT_TS values. The symbol "+" denotes samples that differ significantly, if p is significant at 5% ($p < 0.05$)

PDB ID	**RMSD (Å)**			**GDT (%)**		
	Low.	**Avg.**	p**-value**	**High.**	**Avg.**	p**-value**
1AB1-MA	**2.75**	**4.3** ± (1.59)	6.39e-02	**74.46**	**66.17** ± (6.85)	6.79e-03 +
1AB1-Rosetta	3.45	5.55 ± (1.02)		62.5	56.45 ± (4.27)	
1ACW-MA	1.75	2.41 ± (0.57)	1.59e-01	75.86	71.44 ± (3.35)	1.02e-01
1ACW-Rosetta	**1.66**	**2.11** ± (0.38)		**77.59**	**73.49** ± (3.33)	
1AIL-MA	**3.26**	**3.89** ± (0.45)	4.70e-04 +	**62.14**	**57.72** ± (3.13)	4.65e-04 +
1AIL-Rosetta	6.85	9.45 ± (1.05)		48.93	39.33 ± (5.36)	
1DFN-MA	**2.88**	**3.67** ± (0.62)	2.69e-03 +	**55.0**	**50.42** ± (2.89)	2.21e-03 +
1DFN-Rosetta	3.63	5.29 ± (0.86)		49.17	44.69 ± (2.6)	
1ENH-MA	**1.57**	**2.08** ± (0.43)	2.03e-02 +	44.44	43.34 ± (0.83)	5.00e-01
1ENH-Rosetta	1.7	3.08 ± (0.95)		**46.3**	**43.52** ± (1.27)	
1FNA-MA	7.49	**10.04** ± (1.97)	3.18e-01	**18.13**	**15.28** ± (1.77)	2.81e-01
1FNA-Rosetta	**7.06**	10.45 ± (1.71)		17.86	14.7 ± (1.87)	
1OPD-MA	3.92	**6.79** ± (1.76)	1.57e-02 +	53.53	**43.09** ± (6.38)	2.81e-01
1OPD-Rosetta	**2.74**	9.88 ± (3.25)		**72.06**	43.05 ± (12.82)	
1Q2K-MA	2.14	3.23 ± (0.76)	2.03e-02 +	75.81	65.32 ± (6.4)	1.37e-02 +
1Q2K-Rosetta	**0.57**	**2.03** ± (1.06)		**94.35**	**78.43** ± (11.39)	
1ROP-MA	**1.26**	**1.75** ± (0.36)	6.80e-04 +	**86.16**	**79.07** ± (4.31)	7.78e-03 +
1ROP-Rosetta	1.98	5.27 ± (2.83)		79.91	62.83 ± (12.28)	
1UTG-MA	**2.84**	**3.46** ± (0.53)	1.38e-03 +	**65.71**	**61.96** ± (4.52)	2.66e-03 +
1UTG-Rosetta	4.04	6.96 ± (2.94)		58.93	49.42 ± (9.43)	
2MR9-MA	1.48	**1.76** ± (0.2)	1.14e-01	82.95	**78.55** ± (3.58)	9.41e-02
2MR9-Rosetta	**1.43**	2.22 ± (0.69)		**83.52**	73.79 ± (6.59)	
2P5K-MA	3.63	4.17 ± (0.52)	3.70e-03 +	48.41	46.03 ± (1.43)	9.08e-04 +
2P5K-Rosetta	**1.57**	**2.29** ± (1.0)		**53.97**	**51.54** ± (1.85)	
2PMR-MA	**1.62**	**2.08** ± (0.42)	6.80e-04 +	**50.33**	**46.79** ± (1.7)	9.74e-04 +
2PMR-Rosetta	2.46	3.77 ± (0.79)		45.39	41.28 ± (2.16)	
3V1A-MA	1.25	**1.5** ± (0.24)	3.96e-01	54.69	**53.19** ± (1.15)	3.96e-01
3V1A-Rosetta	**0.7**	2.51 ± (1.9)		**55.21**	51.44 ± (4.63)	
T0820-MA	**5.63**	**7.57** ± (2.17)	6.39e-02	**46.94**	**42.08** ± (3.71)	1.47e-01
T0820-Rosetta	7.34	9.19 ± (1.7)		45.28	39.62 ± (3.71)	
Resume	53.34%(8/15)	80.0%(12/15)	-	53.34%(8/15)	73.34%(11/15)	-

Source: From the author (2022).

similar to experimental structures if they present RMSD values ≤ 4 since the crystallographic structures in PDB show a momentary state of the protein, which is always in motion, according to Carugo (CARUGO, 2003). Thus, it can be seen that the MA framework obtained an average RMSD lower than 4 for 10 out of 15 targets, whereas Rosetta has reached an average RMSD lower than 4 for 7 proteins. Regarding the lowest RMSD values, the MA obtained values lower than 4 for 13 targets, and Rosetta has achieved RMSD values lower than 4 for 11 cases. In addition, results from Table 6.12 are illustrated in the box diagrams of Figure 6.27, with respect to the RMSD measure, for the 8 runs of each method for all targets. The red lines on the plots represent the average RMSD of these runs.

Figure 6.27: Box diagrams obtained from the 8 runs of our proposed MA and the Rosetta, regarding the lowest RMSD value (**y-axis**) reached in each run. The red lines on the plots represent the average RMSD of these runs



Source: From the author (2022).

In comparison with Rosetta, the proposed MA achieved better results concerning the average RMSD values for 12 (80.0%) out of 15 targets, of which for 7 cases, the results differed significantly ($p < 0.05$) according to the *Mann-Whitney* similarity test. We note that the MA obtained average RMSD values with a difference greater than 1 in relation to Rosetta for 10 targets. Moreover, concerning the lowest RMSD values, the MA obtained better results than Rosetta for 8 (53.34%) target proteins.

Analyzing the GDT_TS values shown in Table 6.12, we highlight that the obtained results are similar to the ones mentioned above concerning the RMSD measure. It is observed that our method achieved better results than Rosetta regarding the average GDT_TS for 11 (73.34%) cases, of which for 6 targets, the results differed significantly. Besides, regarding the highest GDT_TS values, the MA also reached better results than Rosetta for 8 (53.34%) proteins.

By analyzing the prediction results for the largest target proteins, it can be noticed that the MA outperformed Rosetta in almost all cases. It is noteworthy that the larger the protein, the greater the difficulty of prediction and packing. For instance, the targets T0820, 1OPD, and 2PMR fold into globular well-packed conformations. Such cases represent larger proteins with compact conformations, where the MA surpassed Rosetta regarding all average values of RMSD and GDT_TS. These outcomes demonstrate the method's ability to pack more compact protein conformations properly.

Furthermore, drawing a parallel with the obtained results from the scenario of multimodal optimization, it is possible to note that the issue regarding the dimensionality of variables, previously observed in that case study, has been reduced in the current scenario. The probable reason for that is due to the knowledge-based strategies incorporated in the algorithm, via the first stage of sampling solutions and the specific-problem evolutionary components, combined with the optimization step focused on multimodal objective functions. As mentioned earlier, these strategies can support the method over the search process by constraining the search space, and it may have better guided the optimization towards promising regions. Hence, such hybridization of components may reduce, to a certain degree, the shortcomings pointed out in the previous scenarios. This combination of strategies has improved the method's performance as the exploration was enhanced and more refined solutions were found. Such results also reinforce the need to include previous knowledge about the problem in the search strategies.

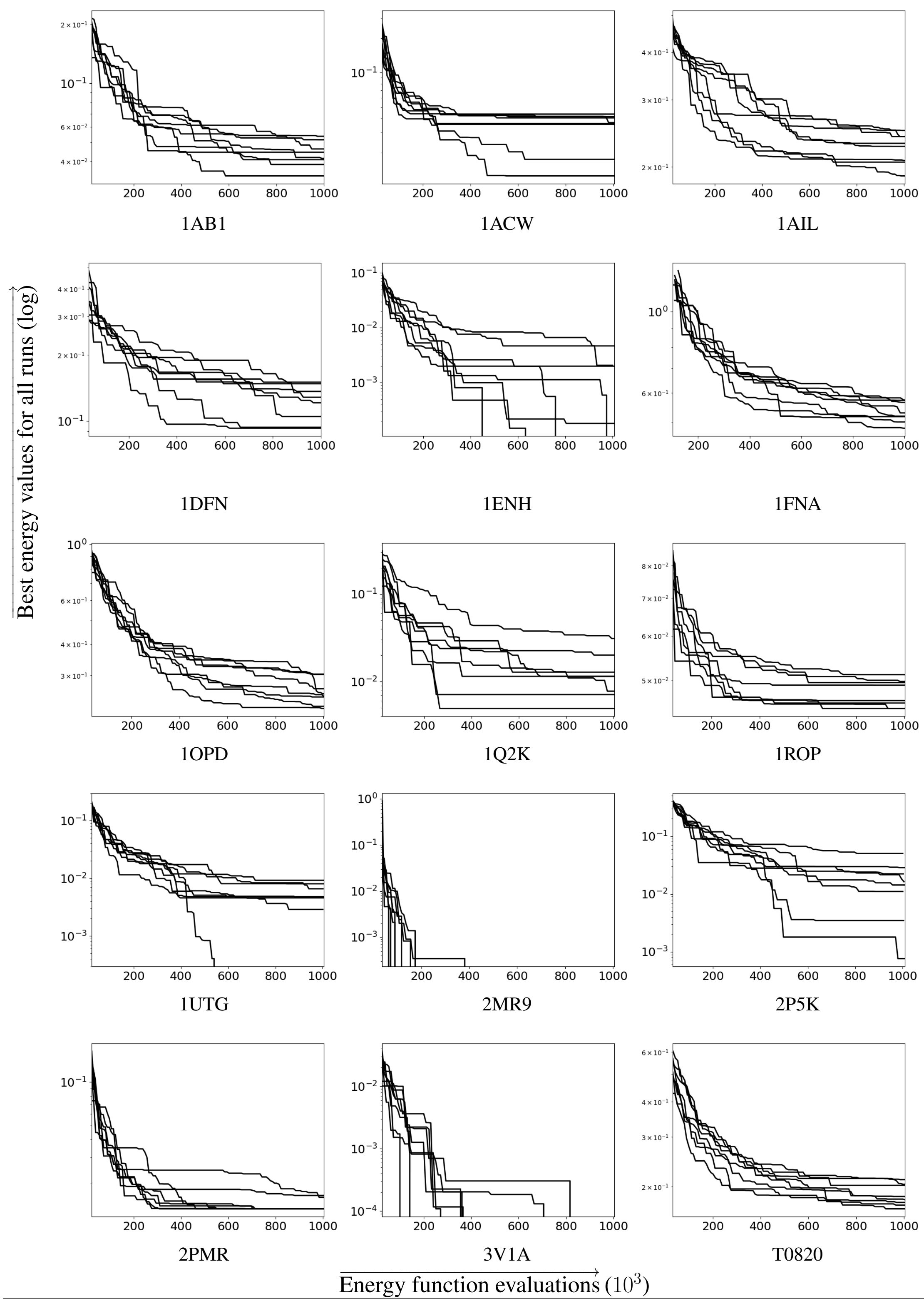
Contrarily, despite the reasonable results for the PSP, the MA could not reach average RMSD values lower than 4 for 4 targets, emphasizing 1FNA, 1OPD, and T0820. These proteins are the largest amino acid sequences in the benchmark set. The main issue regarding these targets is the difficulty in modeling irregular regions of the structure, which are responsible for correctly positioning the protein chains and secondary structures. It indicates that only reaching reasonable structure packing is not enough depending on the intermolecular relationship complexity. In this sense, we understand that our algorithm must evolve with respect to the modeling of irregular structures of larger proteins.

Thereby, to demonstrate the convergence of the MA regarding the obtained energy (fitness) values throughout the runs, Figure 6.28 shows the convergence plots for all targets. It illustrates the convergence of the method concerning the best energy values reached throughout the 8 runs. Besides that, Figure 6.29 demonstrates the algorithm's convergence regarding the best energy values of each node of the tree throughout the 8 runs. Similar to the plots mentioned above, it shows the convergence of the MA but considers each node of the tree. Each color in the plots represents the nodes of each layer of the tree, and each execution of the method could use the maximum of 13 nodes ($N_{nodes} = 13$) as the general MA framework (Section 6.2). For both figures, the y-axis values are expressed in the logarithm scale.

Analyzing the energy convergence plots shown in Figure 6.28, it can be seen that the algorithm was able to achieve a consistent convergence degree among the final best energy values of each execution for almost all targets. It is also highlighted the fast convergence speed for some targets, such as 2MR9 and 3V1A. We believe that this is related to the target length and folding complexity. Besides that, it can be attributed to the sampling and initialization stage, where the best generated solutions are selected as the MA's initial population. Furthermore, from the convergence behavior of each node of the tree shown in Figure 6.29, we notice that despite the algorithm's convergence behavior concerning the lowest energy values, it could preserve the population diversity as represented by the variability of the energy values of the nodes regarding different stages of the optimization. Thus, we believe that this diversification of the subpopulations and the use of niching strategies have contributed to the obtained results for the PSP due to the interactions between distinct regions of the search space with individuals in different degrees of optimization. Such results demonstrate the importance of adapting the method to deal with the multimodality issues of the problem by regulating the population diversity throughout the optimization process.

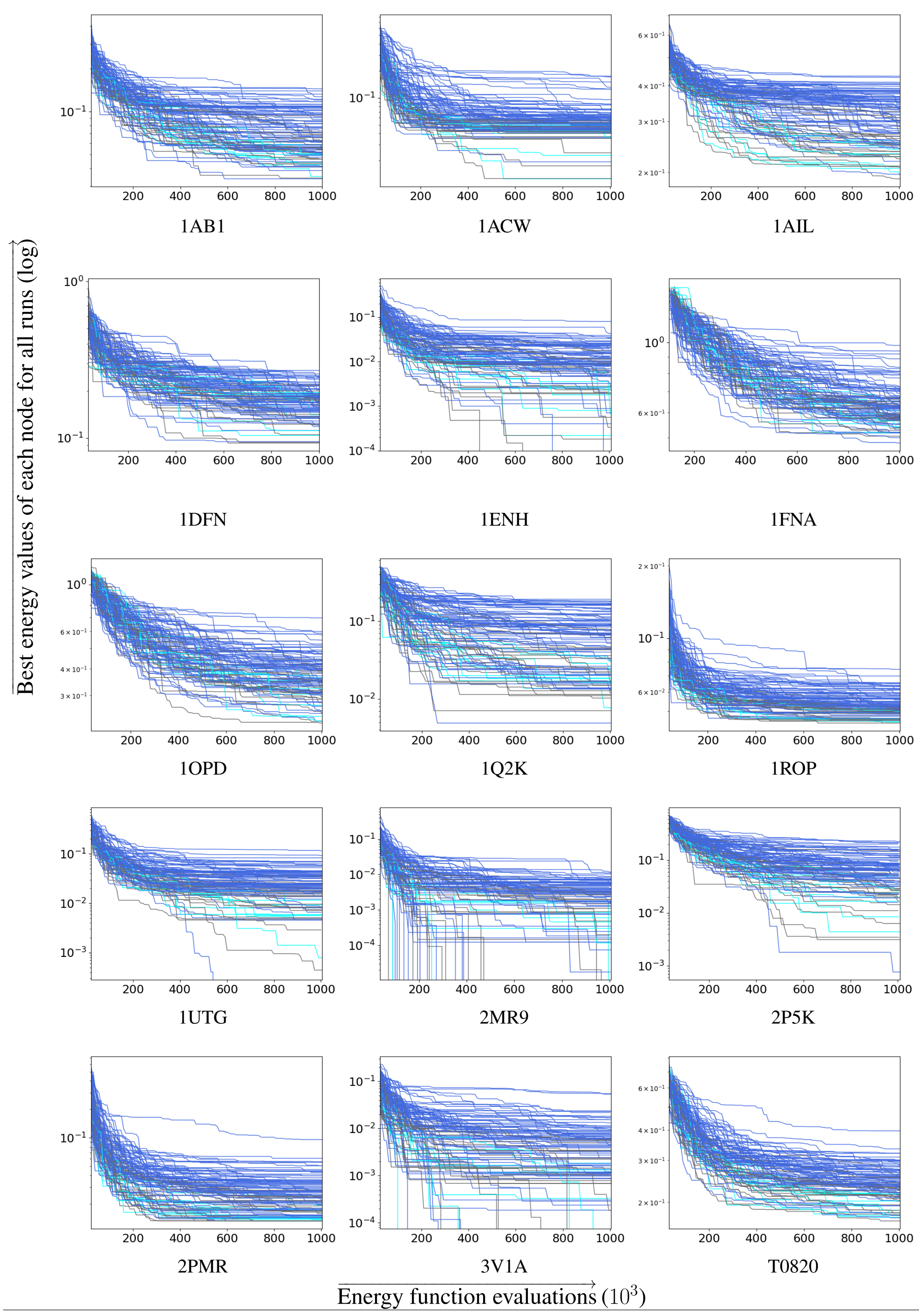
Finally, the best models were superimposed on their corresponding experimental structures to visualize the protein conformations for each target concerning the lowest

Figure 6.28: Convergence of the MA regarding the best energy values (**y-axis**) throughout each of 8 runs (**x-axis**) for the benchmark target proteins



Source: From the author (2022).

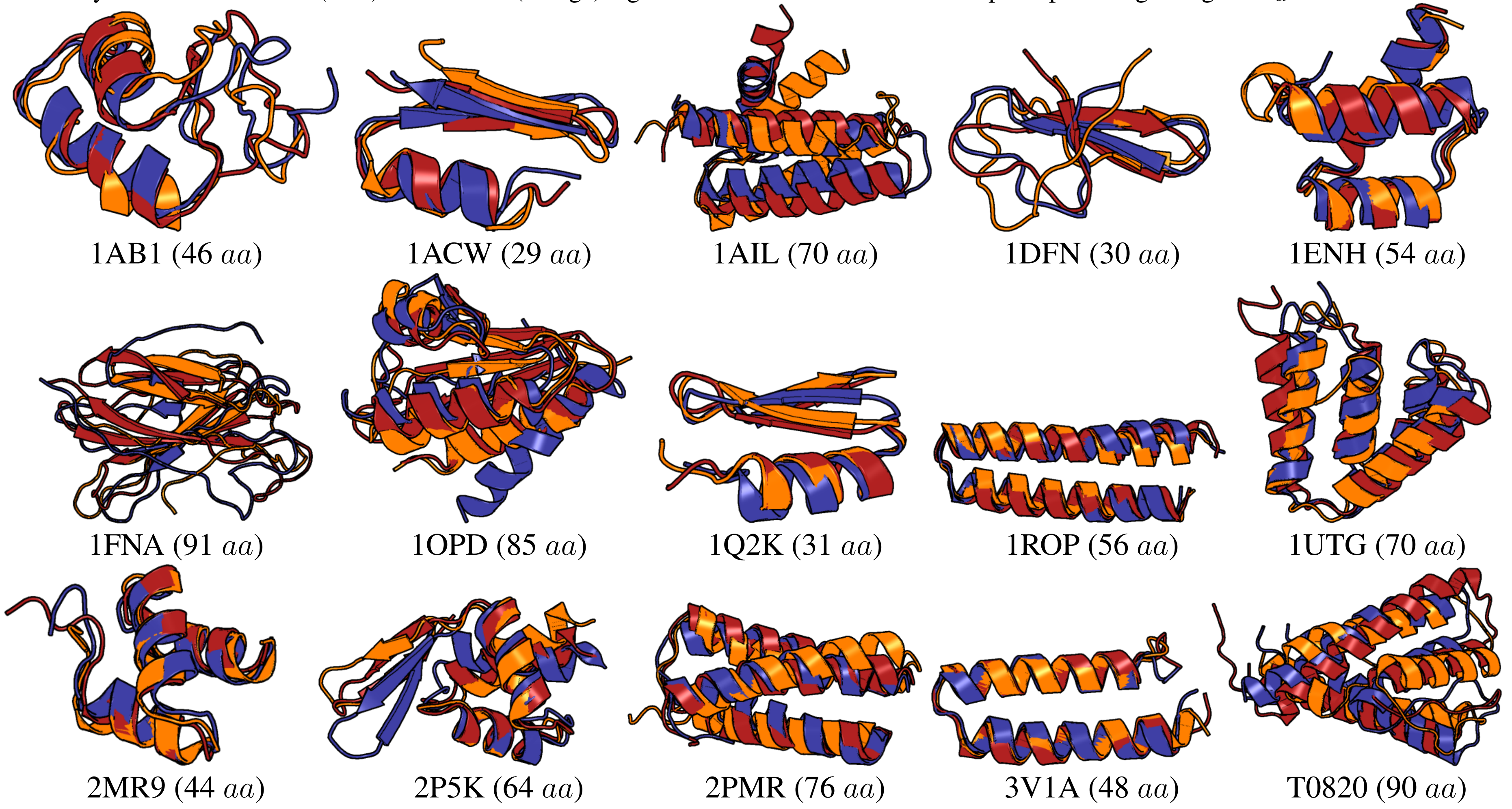
Figure 6.29: Convergence of the MA regarding the best energy values of each node of the tree (**y-axis**) throughout each of 8 runs (**x-axis**) for the benchmark target proteins



Source: From the author (2022).

RMSD values among all runs reached by MA and Rosetta. Thus, Figure 6.30 shows the cartoon superimposed representation of the predicted protein models (lowest RMSD) and the experimentally determined protein structures (red), regarding the MA (blue) and Rosetta (orange).

Figure 6.30: Cartoon superposition representation of experimentally determined protein structures (red) and the models with lowest RMSD values reached by the MA framework (blue) and Rosetta (orange) algorithms. The 3-D models were superimposed regarding the $\mathbb{C}_\alpha$ atoms



Source: From the author (2022).

Analyzing the protein structure representations in Figure 6.30, it is possible to visually verify that the MA has been able to achieve topologies (overall fold in blue) similar to the experimentally determined ones, illustrated in red. We highlight the method's ability to predict packaged conformations according to the experimental structures. The MA also predicted conformations very similar to the ones predicted by Rosetta, and in many cases, our approach has surpassed it, following the results shown in Table 6.12. Nonetheless, despite the package conformations shown in the cartoons, the method could not reach good enough average RMSD and GDT_TS values for the targets 1FNA, 1OPD, and T0820.

Therefore, the obtained results presented in this section demonstrate the prediction potential of the MA framework when compared with the experimental protein structures and the method of Rosetta. However, it is known that the method still needs enhancements, such as the difficulty in modeling irregular structures of larger proteins. With this, it is noted that despite the evolutionary components used in the metaheuristics for the PSP, the need to adapt them to the problem under study is reinforced, aiming to overcome the method weaknesses by the knowledge discovery of specific characteristics concerning the target proteins.

Finally, we believe that the MA framework presented in this work can be seen as an effective initial contribution to the PSP field, but understanding its limitations and required improvements. As further research concerning the ideas presented in this thesis not only for the PSP, it is possible to test the optimization approach on larger and more complex target proteins. Regarding the algorithmic structure of the MA, as delineated in the previous scenarios, it is possible to test the method with distinct parameterization in an attempt to fit the problem better; expand the tree-based structure by regulating the number of nodes and individuals in each subpopulation (node dynamic independence), and reorganizing the interactions between the nodes; investigate new structural metrics and other experimental knowledge sources to be incorporated in the method; improve the knowledge-based components by using refined strategies based on knowledge discovery and pattern recognition from experimentally determined structures to enhance the method's results as a whole.

6.5 Final Remarks

In this chapter, we presented the computational experiments conducted to analyze three delineated scenarios of multimodal continuous optimization and the corresponding implemented algorithms. The scenarios were divided into single global optimization, multimodal optimization with more than one global optimum, and the real problem of predicting the 3-D protein structures. The chapter described the benchmark test functions for each scenario, the algorithms used for comparison, the parameter setting and metrics applied for evaluation, the convergence analyses of the method, and the obtained results regarding the performance of the proposed MA framework for each one of the case studies. It also described the intermediate MA versions and results used to reach the final versions of the framework-based MA applied to each optimization scenario.

Thereby, in general, the MA framework was able to perform well on the above-mentioned scenarios of multimodal optimization by reaching promising results when compared with relevant methods related to the corresponding research fields. Nonetheless, despite the obtained results, we highlight that each designed version of the method still needs improvements in each of these case studies to enhance the obtained results further.

Finally, the next chapter presents the conclusions and final considerations of this work, as well as the future works delineated from the development of this thesis and the presented results.

7 CONCLUSIONS

In spite of the advances in the computational methods and the wide range of metaheuristics proposed for multimodal continuous optimization, there is still the need for developing new strategies focused on issues related to the performance of the algorithms when applied to challenging problems with very complex fitness landscapes (DAS et al., 2011; LI et al., 2016).

In this sense, the general idea of this thesis was centered around the investigation of distinct metaheuristic aspects to deal with problems in the multimodal continuous domain. Within this context, we highlight the points of interest outlined for this work in the conception of the metaheuristics: (*i*) discovery and maintenance of the search space optimal or suboptimal points through multimodal strategies; (*ii*) the balance between Expr/Expt search operations; and (*iii*) the parameter control problem.

To accomplish that, we defined the multimodal PSP problem as our real case study, which is one of the most important problems in Structural Bioinformatics. Thus, we designed the proposed methods from a constructive perspective, starting from a more general optimization approach and benchmark functions for global optimization. We gradually improved the proposed algorithms, changing the functions toward the real problem. With the development of the work, we intended to create, via an incremental approach, a method capable of dealing with the inherent multimodality and issues of a range of optimization functions while trying to preserve accurate results. Our focus was also to evaluate the behavior of the methods facing different multimodal optimization scenarios.

Therefore, we implemented an adaptive MA as a framework-based method with multiple populations, which incorporated concepts of bio-inspired algorithms for global optimization with separate local improvement. It is known that MAs enable the combination of ideas from different search methodologies, which may provide better results than a single strategy for a given problem. Following the points of interest outlined for this thesis, it aimed to explore and refine the search space in an attempt to find and maintain the existing optimal solutions; keep a reasonable balance between Expr/Expt mechanisms through population diversity measures; and control the parameter setting of the method via dynamic strategies to enhance its performance.

Thus, we designed three versions of the MA framework for three different scenarios of multimodal optimization: (*i*) the general MA framework for single global continuous optimization with multimodal fitness function; (*ii*) the MA framework with archive strategy for multimodal optimization with more than one global optimum; and (*iii*) the MA framework with specific-problem components for the multimodal problem of predicting the 3-D protein structures.

The general MA framework presented in the first scenario was used as the basis for developing the other designed versions employed to deal with the subsequent problems. The algorithm was implemented as a general optimization framework that is easily adapted to other search strategies and components. It has been shown by the intermediate versions implemented to reach the final version for global optimization. The method was structured based on a hierarchical tree data structure implemented under an adaptive MA framework. It consists of a multi-population technique, which arranges the population of individuals in subpopulations into the tree's nodes through a clustering-based niching strategy. The method included independent intra-node optimization through a core metaheuristic and interactions between distinct subpopulations as a way of knowledge sharing, population diversification, and exploration inter and intra niches. Besides that, a modified hybrid-ABC algorithm was proposed as a core search method, where each node of the tree performs an independent execution. The framework also incorporated adaptive and hybrid approaches for local improvements, population convergence and performance, and dynamic parameter adaptation. Thus, this first version was tested on the 100-Digit Challenge on single objective real-parameter optimization[1] (PRICE et al., 2018) adopted in the CEC 2019, GECCO 2019 and SEMCCO 2019 (PRICE et al., 2019).

The second designed version of the MA was modified to focus on multimodal optimization when dealing with objective functions with more than one global optimum. The main issues considered in such a variant are related to discovering and maintaining the existing global optima of a given problem while locating new ones throughout the search process. This version incorporated an external archive to store the identified global optima when converged regions of the search space are detected through the MA control procedure for convergence and performance. This version was tested on the benchmark functions of the CEC 2013 for multimodal function optimization[2] [3] (LI; ENGELBRECHT; EPITROPAKIS, 2013).

Lastly, the MA for the PSP was enhanced with the problem-dependencies and data knowledge from experimentally determined 3-D protein structures to turn it more robust to the problem. The method was divided into two main optimization stages. The

[1]<https://github.com/P-N-Suganthan/CEC2019>
[2]<http://epitropakis.co.uk/cec20-niching/competition/>
[3]<http://epitropakis.co.uk/gecco2020/>

first step of sampling and initializing structural models is responsible for generating and classifying several models from the amino acid primary sequence of a target protein. It aimed to reduce the search space size, overcome its roughness and provide feasible initial solutions to the second optimization step. The second step represents the optimization of solutions sampled from the first stage. The MA framework performs the search process, which was incremented to deal with the problem. The method included modified specific-problem operators, such as the Uniform SS crossover and the constraints in the mutation operations based on conformational preferences of amino acids. Such strategies aimed to increase the method's performance and the accuracy of the solutions, focusing on the characteristics of the PSP. This method variant was applied to the PSP problem through a test set of 15 target proteins with distinct sequences, sizes, and topologies.

Thereby, it is possible to summarize the obtained results regarding the MA framework's performance facing each of the optimization scenarios as follows.

Results of the global optimization scenario: The obtained results of the general MA for global optimization demonstrated that our method could reach reasonable results for almost all functions, including the hardest ones. Moreover, the MA has reached a consistent convergence degree concerning the fitness values over the search process for all benchmark functions. Following the points of interest outlined for this thesis, it is observed that the algorithm could balance the Expr/Expt efforts and preserve the population diversity while consistently improving the solutions.

In general, through the search components implemented in this version of the MA, it was able to optimize the benchmark functions by exploring and refining the search space, as well as overcoming premature convergence to local optima in many scenarios.

Nonetheless, the results have also indicated that the algorithm still needs improvements, mainly concerning the parameter control, i.e., an additional effort should be made to adapt the parameter setting to any function properly. For instance, it is possible to include additional parameter control strategies not to have to tune the method for a specific benchmark function and better orchestrate all framework components throughout the search process.

Results concerning the case study of multimodal optimization: By analyzing results obtained from benchmark functions for multimodal optimization with more than one global optimum, we highlight that the MA was able to perform well on test functions considering $\epsilon = 1.0E - 01$, which was measured via peak ratio and success rate. The method has reached PR values higher than 90% for 18 out of 20 functions, which may indicate reasonable exploration ability to locate and preserve optimal solutions. Besides that, the MA was able to control the Expr/Expt efforts over the optimization process and has converged to optimal regions while maintaining population diversity to find more distinct global optima, which is one of the main issues of multimodal optimization.

Nonetheless, the algorithm obtained PR values higher than 90% for only 9 out of 20 functions considering $\epsilon = 1.0E - 05$, which means the highest level of accuracy. Moreover, the MA could not perform well on the problem with the highest dimensionality among all other functions. Such results may indicate that the algorithm was able to find the global optima's neighborhood for the hardest cases. However, it could not correctly exploit these found optima to achieve more accurate solutions.

In general, the MA framework performed well on the tested benchmark suite, except for the pointed out issues. Thus, the MA still needs improvements in certain aspects, such as exploitation of solutions and dimensionality of variables. To overcome these issues in future works, one could enhance the exploitation ability of the MA by focusing more on LS by regulating the LIP procedure for this purpose. Another improvement to the MA for overall performance could consist of including components for high-dimensional optimization to overcome the multimodality issues of larger search spaces; using a more robust archive strategy to deal with the found optimal solutions; and testing a distinct number of layers in the tree since a more robust archive strategy may enhance the method's abilities of Expr/Expt.

Results concerning the prediction of 3-D protein structures: The experiments in the last scenario aimed to evaluate the method's potential when tackling a complex real-world multimodal problem and compare it to the method of Rosetta (ROHL et al., 2004; SONG et al., 2013), which is one of the state-of-the-art algorithms for PSP. The structural quality of the predicted structures was evaluated by similarity comparisons with experimentally determined ones regarding the RMSD and GDT_TS structural measures.

According to the obtained prediction results, the proposed MA framework was able to outperform the method of Rosetta in many target proteins. The algorithm has also predicted 3-D protein structures with similar topologies (overall fold) to the experimentally determined ones. Briefly, our method achieved better results than Rosetta, concerning the average RMSD and GDT_TS values, for 12 (80.0%) and 11 (73.34%) out of 15 targets, of which for 7 and 6 cases, the results differed significantly ($p < 0.05$) according to the *Mann-Whitney* similarity test, respectively. The results also demonstrated the method's ability to predict packaged conformations according to the experimental structures properly. For instance, it was noticed that the MA outperformed Rosetta in all cases concerning the largest target proteins.

Hence, the method was able to reduce the effects of the issue regarding the dimensionality of variables previously observed in the scenario of multimodal optimization. The probable reason for that is due to the knowledge-based strategies incorporated in the algorithm combined with the search framework focused on multimodal objective functions. In addition, the algorithm presented a consistent convergence degree for almost all target proteins. Despite this observed convergence behavior for the lowest energy values found, the method could also preserve the population diversity among the tree's nodes regarding different stages of the optimization. Such results reinforce the importance of adapting the method to deal with the multimodality issues of the functions by including previous knowledge about the problem understudy in the search strategies and regulating the Expr/Expt mechanisms throughout the optimization process.

Finally, we can state that the proposed MA framework designed as an incremental algorithm via the combination of distinct search strategies to address multimodal objective functions can be seen as an effective initial contribution not only to the PSP problem but also to the multimodal continuous optimization field. We highlight that the concepts and search algorithms proposed in this work can be extended to any other optimization problem that follows the same principles of optimization already described. However, we emphasize that the method still needs improvements so that the obtained results can be enhanced as a whole for the different scenarios of optimization. For instance, we observed that the MA could not reach good enough results for some larger targets due to the difficulty of modeling irregular protein structure regions. Thus, one can try to overcome such an issue and improve the method through knowledge discovery of specific characteristics of the target proteins.

7.1 Research Perspectives and Future Works

This thesis work points out interesting research guidelines concerning metaheuristics and multimodal optimization when dealing with each one of the idealized scenarios of multimodal optimization. As already discussed from the obtained results, the different versions of the MA framework still need some improvements in each case study to enhance the method's performance and results as a whole.

Therefore, the work also raises interesting research topics to be explored in this field, with relevant multidisciplinary applications in Computer Science and Structural Bioinformatics. With this, as further research concerning the ideas presented in this work, it is possible to delineate some perspectives of work as a continuation of this work:

- To incorporate in the framework additional parameter control strategies to not have to tune the method for a specific benchmark function and better orchestrate the all components of the framework throughout the optimization process, such as the number of layers, nodes, and individuals in the tree structure or the threshold used to regulate the Expr/Expt mechanisms;
- To enhance the exploitation ability of the MA by focusing more on LS by regulating the LIP procedure for this purpose in order to properly exploit the hardest multimodal landscapes and achieve more accurate solutions;
- To incorporate in the framework components for high-dimensional optimization to overcome the multimodality issues of larger search spaces;
- To implement a more robust archive strategy to deal with the found optimal solutions over the search process;
- To test different configurations of tree data structure, such as the number of layers in the tree and the number of individuals in each node, since a more robust archive strategy may enhance the method's abilities of Expr/Expt;
- Following the previous topic, to test the method with distinct parameterization in an attempt to better fit the problems;
- To test other search strategies and hybridizations in an attempt to improve the obtained results for all scenarios;
- Specifically for the PSP problem, to investigate other experimental knowledge sources to be incorporated in the method as a way to guide the prediction process better;
- To improve the knowledge-based components through the use of strategies based on knowledge discovery and pattern recognition from experimentally determined structures to enhance the method's results;
- To explore novel characteristics of the target proteins through knowledge discovery to overcome the difficulty of the method in modeling irregular regions of the protein structures;
- To test the proposed method on distinct benchmark functions for multimodal optimization and larger and more complex target proteins for the PSP.

7.2 Publications

This section presents the works developed during the Ph.D. course and previous relevant ones, covering the research fields of artificial intelligence and optimization, meta-heuristics, and 3-D protein structure prediction.

Articles

- **CORRÊA, L. L.**; DORN, M. A multi-population memetic algorithm for the 3-D protein structure prediction problem. Swarm and Evolutionary Computation, v. 55, n. 100677, p. 1–36, 2020.

- INOSTROZA-PONTA, M.; DORN, M.; ESCOBAR, I.; **CORRÊA, L. L.**; ROSAS, E.; HIDALGO, N.; MARIN, M. Exploring the high selectivity of 3-D protein structures using distributed memetic algorithms. Journal of Computational Science, v. 41, n. 101087, p. 1–17, 2020.

- **CORRÊA, L. L.**; BORGUESAN, B.; KRAUSE, M. J.; DORN, M. Three-Dimensional Protein Structure Prediction Based on Memetic Algorithms. Computers & Operations Research, v. 91, p. 160-177, 2018.

Conference Proceedings

- **CORRÊA, L. L.**; ARANTES, L.; SENS, P.; INOSTROZA-PONTA, M.; DORN, M. A dynamic evolutionary multi-agent system to predict the 3D structure of proteins. In: IEEE Congress on Evolutionary Computation, 2020, Glasgow, UK. Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC 2020), 2020. p. 1-8.

- **CORRÊA, L. L.**; DORN, M. A Multi-objective Swarm-Based Algorithm for the Prediction of Protein Structures. In: International Conference on Computational Science - ICCS 2019, Faro, Portugal. Lecture Notes in Computer Science. 1ed.:

Springer, Cham, 2019, v. 11538, p. 101-115.

- **CORRÊA, L. L.**; DORN, M. A knowledge-based artificial bee colony algorithm for the 3-D protein structure prediction problem. In: IEEE Congress on Evolutionary Computation, 2018, Rio de Janeiro, Brazil. Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC 2018), 2018. p. 1-8.

- **CORRÊA, L. L.**; INOSTROZA-PONTA, M.; DORN, M. An evolutionary multi-agent algorithm to explore the high degree of selectivity in three-dimensional protein structures. In: IEEE Congress on Evolutionary Computation, 2017, Donostia, Spain. Proceedings of the IEEE Congress on Evolutionary Computation (IEEE CEC 2017), 2017. p. 1111-1118.

REFERENCES

ABRIATA, L. A. et al. Assessment of hard target modeling in casp12 reveals an emerging role of alignment-based contact prediction methods. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 86, p. 97–112, 2018.

ADHIKARI, B.; HOU, J.; CHENG, J. Protein contact prediction by integrating deep multiple sequence alignments, coevolution and machine learning. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 86, p. 84–96, 2018.

AHRARI, A.; DEB, K.; PREUSS, M. **Benchmarking covariance matrix self adaption evolution strategy with repelling subpopulations for GECCO 2017 competition on multimodal optimization**. USA, Germany, 2017. 5 p.

AHRARI, A.; DEB, K.; PREUSS, M. Multimodal optimization by covariance matrix self-adaptation evolution strategy with repelling subpopulations. **Evol. Comput.**, MIT Press, v. 25, n. 3, p. 439–471, 2017.

AKAY, B.; KARABOGA, D. A modified artificial bee colony algorithm for real-parameter optimization. **Inf. Sci.**, Elsevier, v. 192, p. 120–142, 2012.

ALATAS, B. Chaotic bee colony algorithms for global numerical optimization. **Expert Syst. Appl.**, Elsevier, v. 37, n. 8, p. 5682–5687, 2010.

ALETI, A.; MOSER, I. A systematic literature review of adaptive parameter control methods for evolutionary algorithms. **ACM Comput. Surv.**, ACM New York, v. 49, n. 3, p. 1–35, 2016.

ALETI, A. et al. Choosing the appropriate forecasting model for predictive parameter control. **Evol. Comput.**, MIT Press, v. 22, n. 2, p. 319–349, 2014.

ANAM, S. Multimodal optimization by using hybrid of artificial bee colony algorithm and bfgs algorithm. **J. Phys.: Conf. Ser.**, IOP Publishing, v. 893, n. 012068, p. 1–8, 2017.

ANFINSEN, C. B. Principles that govern the folding of protein chains. **Science**, American Association for the Advancement of Science, v. 181, n. 4096, p. 223–230, 1973.

ANFINSEN, C. B. et al. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. **Proc. Natl. Acad. Sci. USA**, National Academy of Sciences, v. 47, n. 9, p. 1309–1314, 1961.

AUER, P.; CESA-BIANCHI, N.; FISCHER, P. Finite-time analysis of the multiarmed bandit problem. **Mach. Learn.**, Springer, v. 47, n. 2-3, p. 235–256, 2002.

AWAD, N. H. et al. **Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization**. Singapore, Jordan, China, 2016. 34 p.

BACK, T. **Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms**. 1. ed. Oxford, UK: Oxford University Press, 1996. 328 p.

BAXEVANIS, A. D.; OUELLETTE, B. F. **Bioinformatics: a practical guide to the analysis of genes and proteins**. 2. ed. New York, USA: John Wiley & Sons, Inc., 2004. 495 p.

BEDEIAN, A. G.; MOSSHOLDER, K. W. On the use of the coefficient of variation as a measure of diversity. **Organ. Res. Methods**, Sage Publications, v. 3, n. 3, p. 285–297, 2000.

BEHESHTI, Z.; SHAMSUDDIN, S. M. H. A review of population-based meta-heuristic algorithms. **Int. J. Adv. Soft Comput. Appl**, v. 5, n. 1, p. 1–35, 2013.

BELDA, I. et al. Evolutionary computation and multimodal search: A good combination to tackle molecular diversity in the field of peptide design. **Mol. Diversity**, Springer, v. 11, n. 1, p. 7–21, 2007.

BERMAN, H. M. et al. The protein data bank. **Nucleic Acids Res.**, Oxford University Press, v. 28, n. 1, p. 235–242, 2000.

BEYER, H.-G.; SENDHOFF, B. Covariance matrix adaptation revisited – the cmsa evolution strategy –. In: INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE. **Proceedings...** Dortmund, Germany: Springer, 2008. p. 123–132.

BORGUESAN, B.; INOSTROZA-PONTA, M.; DORN, M. Nias-server: Neighbors influence of amino acids and secondary structures in proteins. **J. Comput. Biol.**, Mary Ann Liebert, Inc., v. 24, n. 3, p. 255–265, 2016.

BORGUESAN, B. et al. Apl: An angle probability list to improve knowledge-based metaheuristics for the three-dimensional protein structure prediction. **Comput. Biol. Chem.**, Elsevier, v. 59, p. 142–157, 2015.

BOSMAN, P. A.; GRAHL, J.; THIERENS, D. Benchmarking parameter-free amalgam on functions with and without noise. **Evol. Comput.**, MIT Press, v. 21, n. 3, p. 445–469, 2013.

BOUSSAÏD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. **Inf. Sci.**, Elsevier, v. 237, p. 82–117, 2013.

BOWIE, J. U.; LUTHY, R.; EISENBERG, D. A method to identify protein sequences that fold into a known three-dimensional structure. **Science**, American Association for the Advancement of Science, v. 253, n. 5016, p. 164–170, 1991.

BRADLEY, P.; MISURA, K. M.; BAKER, D. Toward high-resolution de novo structure prediction for small proteins. **Science**, American Association for the Advancement of Science, v. 309, n. 5742, p. 1868–1871, 2005.

BRANDEN, C.; TOOZE, J. **Introduction to protein structure**. 2. ed. New York, USA: Garland Science, 1999. 410 p.

BREST, J. et al. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. **IEEE Trans. Evol. Comput**, IEEE, v. 10, n. 6, p. 646–657, 2006.

BREST, J.; MAUČEC, M. S.; BOŠKOVIĆ, B. Single objective real-parameter optimization: Algorithm jso. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Donostia, Spain: IEEE, 2017. p. 1311–1318.

BREST, J.; MAUČEC, M. S.; BOŠKOVIĆ, B. The 100-digit challenge: Algorithm jde100. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Wellington, New Zealand: IEEE, 2019. p. 19–26.

CAI, Q. et al. Enhancing artificial bee colony algorithm with dynamic best neighbor-guided search strategy. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Glasgow, UK: IEEE, 2020. p. 1–8.

CALLAWAY, E. 'it will change everything': Deepmind's ai makes gigantic leap in solving protein structures. **Nature**, Nature Research, v. 588, p. 203–204, 2020.

CAO, Y. et al. An improved global best guided artificial bee colony algorithm for continuous optimization problems. **Clust. Comput.**, Springer, v. 22, n. 2, p. 3011–3019, 2019.

CARUGO, O. How root-mean-square distance (r.m.s.d.) values depend on the resolution of protein structures that are compared. **J. Appl. Crystallogr.**, International Union of Crystallography, v. 36, p. 125–128, 2003.

CAVANAGH, J. et al. **Protein NMR spectroscopy: principles and practice**. 2. ed. New York, USA: Academic Press, 2006. 912 p.

CHAUDHURY, S.; LYSKOV, S.; GRAY, J. Pyrosetta: a script-based interface for implementing molecular modeling algorithms using rosetta. **Bioinformatics**, Oxford University Press, v. 26, n. 5, p. 689–691, 2010.

CHEN, X.; TIANFIELD, H.; LI, K. Self-adaptive differential artificial bee colony algorithm for global optimization problems. **Swarm Evol. Comput.**, Elsevier, v. 45, p. 70–91, 2019.

CHEN, Z.-G. et al. Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems. **IEEE Trans. Evol. Comput.**, IEEE, v. 24, n. 4, p. 708–719, 2019.

CHENG, S. et al. Population diversity maintenance in brain storm optimization algorithm. **J. Artif. Intell. Soft Comput. Res.**, Sciendo, v. 4, n. 2, p. 83–97, 2014.

CHIVIAN, D. et al. Ab initio methods. In: **Structural Bioinformatics**. New Jersey, USA: John Wiley & Sons, Inc, 2003. v. 44, chp. 27, p. 547–557.

CHOU, K.-C. Structural bioinformatics and its impact to biomedical science. **Curr. Med. Chem.**, Bentham Science Publishers, v. 11, n. 16, p. 2105–2134, 2004.

CHOU, K.-C.; ZHANG, C.-T. Prediction of protein structural classes. **Crit. Rev. Biochem. Mol. Biol.**, Taylor & Francis, v. 30, n. 4, p. 275–349, 1995.

COMBS, S. et al. Small-molecule ligand docking into comparative models with rosetta. **Nat. Protoc.**, Nature Research, v. 8, n. 7, p. 1277–1298, 2013.

CONTE, L. L. et al. Scop: a structural classification of proteins database. **Nucleic Acids Res.**, Oxford University Press, v. 28, n. 1, p. 257–259, 2000.

COOK, S. A. An overview of computational complexity. **Commun. ACM**, ACM, v. 26, n. 6, p. 400–408, 1983.

CORRÊA, L. et al. A dynamic evolutionary multi-agent system to predict the 3d structure of proteins. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Glasgow, UK: IEEE, 2020. p. 1–8.

CORRÊA, L. et al. A memetic algorithm for 3-D protein structure prediction problem. **IEEE/ACM Trans. Comput. Biol. Bioinf.**, IEEE, v. 15, n. 3, p. 690–704, 2016.

CORRÊA, L. D. L.; DORN, M. A knowledge-based artificial bee colony algorithm for the 3-d protein structure prediction problem. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Rio de Janeiro, Brazil: IEEE, 2018. p. 1–8.

CORRÊA, L. de L. et al. Three-dimensional protein structure prediction based on memetic algorithms. **Computers & Operations Research**, Elsevier, v. 91, p. 160–177, 2018.

CORRÊA, L. de L.; DORN, M. Multi-agent systems in three-dimensional protein structure prediction. In: **Multi-Agent-Based Simulations Applied to Biological and Environmental Systems**. 1. ed. Hershey PA, USA: IGI Global, 2017. chp. 11, p. 241–278.

CORRÊA, L. de L.; DORN, M. A multi-objective swarm-based algorithm for the prediction of protein structures. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE. **Computational Science - ICCS 2019**. Faro, Portugal: Springer, Cham, 2019, (Lecture Notes in Computer Science, v. 11538). p. 101–115.

CORRÊA, L. de L.; DORN, M. A multi-population memetic algorithm for the 3-d protein structure prediction problem. **Swarm Evol. Comput.**, Elsevier, v. 55, n. 100677, p. 1–36, 2020.

CORRÊA, L. de L.; INOSTROZA-PONTA, M.; DORN, M. An evolutionary multi-agent algorithm to explore the high degree of selectivity in three-dimensional protein structures. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Donostia, Spain: IEEE, 2017. p. 1111–1118.

CREIGHTON, T. E. Protein folding. **Biochem. J.**, Portland Press Ltd, v. 270, n. PMC1131670, p. 1–16, 1990.

ČREPINŠEK, M.; LIU, S.-H.; MERNIK, M. Exploration and exploitation in evolutionary algorithms: A survey. **ACM Comput. Surv.**, ACM, v. 45, n. 3, p. 1–33, 2013.

CRESCENZI, P. et al. On the complexity of protein folding. **J. Comput. Biol.**, Mary Ann Liebert, Inc., v. 5, n. 3, p. 423–465, 1998.

CUI, L. et al. A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation. **Inf. Sci.**, Elsevier, v. 367, p. 1012–1044, 2016.

CUI, L. et al. A novel artificial bee colony algorithm with an adaptive population size for numerical function optimization. **Inf. Sci.**, Elsevier, v. 414, p. 53–67, 2017.

CUTELLO, V.; NARZISI, G.; NICOSIA, G. A multi-objective evolutionary approach to the protein structure prediction problem. **J. R. Soc. Interface**, The Royal Society, v. 3, n. 6, p. 139–151, 2006.

DAS, S. et al. Real-parameter evolutionary multimodal optimization-a survey of the state-of-the-art. **Swarm Evol. Comput.**, Elsevier, v. 1, n. 2, p. 71–88, 2011.

DAS, S.; MULLICK, S. S.; SUGANTHAN, P. N. Recent advances in differential evolution–an updated survey. **Swarm Evol. Comput.**, Elsevier, v. 27, p. 1–30, 2016.

DAS, S.; SUGANTHAN, P. N. Differential evolution: A survey of the state-of-the-art. **IEEE Trans. Evol. Comput**, IEEE, v. 15, n. 1, p. 4–31, 2010.

DAWKINS, R. **The selfish gene**. 1. ed. Oxford, UK: Oxford university press, 1976. 224 p.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Trans. Evol. Comput.**, IEEE, v. 6, n. 2, p. 182–197, 2002.

DILL, K. A.; MACCALLUM, J. L. The protein-folding problem, 50 years on. **Science**, American Association for the Advancement of Science, v. 338, n. 6110, p. 1042–1046, 2012.

DOKEROGLU, T. et al. A survey on new generation metaheuristic algorithms. **Comput. Ind. Eng.**, Elsevier, v. 137, p. 106040, 2019.

DORN, M.; BURIOL, L. S.; LAMB, L. C. A hybrid genetic algorithm for the 3-d protein structure prediction problem using a path-relinking strategy. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** New Orleans, LA, USA: IEEE, 2011. p. 2709–2716.

DORN, M. et al. A knowledge-based genetic algorithm to predict three-dimensional structures of polypeptides. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Cancun, Mexico: IEEE, 2013. p. 1233–1240.

DORN, M. et al. Three-dimensional protein structure prediction: methods and computational strategies. **Comput. Biol. Chem.**, Elsevier, v. 53, p. 251–276, 2014.

DRÉO, J. et al. **Metaheuristics for hard optimization: methods and case studies**. 1. ed. USA: Springer Science & Business Media, 2006. 326 p.

DUBOIS, A.; DEHOS, J.; TEYTAUD, F. Upper confidence tree for planning restart strategies in multi-modal optimization. **Soft Comput.**, Springer, v. 25, n. 2, p. 1007–1015, 2021.

DUNBRACK, R. L.; COHEN, F. E. Bayesian statistical analysis of protein side-chain rotamer preferences. **Protein Sci.**, Wiley Online Library, v. 6, n. 8, p. 1661–1681, 1997.

DUNKER, A. K. et al. Intrinsically disordered protein. **J. Mol. Graphics Modell.**, Elsevier, v. 19, n. 1, p. 26–59, 2001.

DUNKER, A. K. et al. The unfoldomics decade: an update on intrinsically disordered proteins. **BMC Genom.**, BioMed Central Ltd, v. 9, n. 2, p. 1–26, 2008.

DUNKER, A. K. et al. Function and structure of inherently disordered proteins. **Curr. Opin. Struct. Biol.**, Elsevier, v. 18, n. 6, p. 756–764, 2008.

EIBEN, A. E.; MARCHIORI, E.; VALKO, V. Evolutionary algorithms with on-the-fly population size adjustment. In: INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE. **Proceedings...** Birmingham, UK: Springer, 2004. p. 41–50.

ELOFSSON, A.; GRAND, S. M. L.; EISENBERG, D. Local moves: An efficient algorithm for simulation of protein folding. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 23, n. 1, p. 73–82, 1995.

ELTAEIB, T.; MAHMOOD, A. Differential evolution: A survey and analysis. **Appl. Sci.**, Multidisciplinary Digital Publishing Institute, v. 8, n. 10, p. 1945, 2018.

EPITROPAKIS, M. G.; LI, X.; BURKE, E. K. A dynamic archive niching differential evolution algorithm for multimodal optimization. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Cancun, Mexico: IEEE, 2013. p. 79–86.

FARAGGI, E.; KLOCZKOWSKI, A. A global machine learning based scoring function for protein structure prediction. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 82, n. 5, p. 752–759, 2014.

FONSECA, R.; PALUSZEWSKI, M.; WINTER, P. Protein structure prediction using bee colony optimization metaheuristic. **J. Math. Model. Algo.**, Springer, v. 9, n. 2, p. 181–194, 2010.

FOX, N. K.; BRENNER, S. E.; CHANDONIA, J.-M. The value of protein structure classification information—surveying the scientific literature. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 83, n. 11, p. 2025–2038, 2015.

FREY, B. J.; DUECK, D. Clustering by passing messages between data points. **Science**, American Association for the Advancement of Science, v. 315, n. 5814, p. 972–976, 2007.

GAO, H. et al. An improved artificial bee colony algorithm with its application. **IEEE Trans. Industr. Inform.**, IEEE, v. 15, n. 4, p. 1853–1865, 2018.

GAO, W.; LIU, S.; HUANG, L. A global best artificial bee colony algorithm for global optimization. **J. Comput. Appl. Math.**, Elsevier, v. 236, n. 11, p. 2741–2753, 2012.

GAO, W.; YEN, G. G.; LIU, S. A cluster-based differential evolution with self-adaptive strategy for multimodal optimization. **IEEE Trans. Cybern.**, IEEE, v. 44, n. 8, p. 1314–1327, 2013.

GAO, W.-F. et al. Artificial bee colony algorithm based on information learning. **IEEE Trans. Cybern.**, IEEE, v. 45, n. 12, p. 2827–2839, 2015.

GAO, W.-f.; LIU, S.-y.; HUANG, L.-l. A novel artificial bee colony algorithm based on modified search equation and orthogonal learning. **IEEE Trans. Cybern.**, IEEE, v. 43, n. 3, p. 1011–1024, 2013.

GARZA-FABRE, M. et al. Generating, maintaining and exploiting diversity in a memetic algorithm for protein structure prediction. **Evol. Comput.**, MIT Press, v. 24, n. 4, p. 577–607, 2016.

GINLEY, B. M. et al. Maintaining healthy population diversity using adaptive crossover, mutation, and selection. **IEEE Trans. Evol. Comput.**, IEEE, v. 15, n. 5, p. 692–714, 2011.

GLIBOVETS, N.; GULAYEVA, N. A review of niching genetic algorithms for multimodal function optimization. **Cybern. Syst. Anal.**, Springer US, v. 49, n. 6, p. 815–820, 2013.

GOLDBERG, D. E.; RICHARDSON, J. et al. Genetic algorithms with sharing for multimodal function optimization. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS AND THEIR APPLICATION. **Proceedings...** New York, USA: L. Erlbaum Associates Inc., 1987. p. 41–49.

GUNASEKARAN, K. et al. Extended disordered proteins: targeting function with less scaffold. **Trends Biochem. Sci.**, Elsevier, v. 28, n. 2, p. 81–85, 2003.

GUPTA, D.; GHAFIR, S. An overview of methods maintaining diversity in genetic algorithms. **Int. J. Emerging Technol. Adv. Eng.**, International Journal of Emerging Technology and Advanced Engineering, v. 2, n. 5, p. 56–60, 2012.

GUYEUX, C. et al. Is protein folding problem really a np-complete one? first investigations. **J. Bioinf. Comput. Biol.**, World Scientific, v. 12, n. 01, p. 1350017, 2014.

HALIM, A. H.; ISMAIL, I.; DAS, S. Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. **Artif. Intell. Rev.**, Springer, p. 1–87, 2020.

HANDL, J.; LOVELL, S. C.; KNOWLES, J. Investigations into the effect of multiobjectivization in protein structure prediction. In: INTERNATIONAL CONFERENCE ON PARALLEL PROBLEM SOLVING FROM NATURE. **Proceedings...** Dortmund, Germany: Springer, 2008. p. 702–711.

HAO, M.-H.; SCHERAGAT, H. A. Designing potential energy functions for protein folding. **Curr. Opin. Struct. Biol.**, Elsevier, v. 9, n. 2, p. 184–188, 1999.

HARIK, G. R. Finding multimodal solutions using restricted tournament selection. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. **Proceedings...** San Francisco, USA: Morgan Kaufmann Publishers Inc., 1995. p. 24–31.

HEINIG, M.; FRISHMAN, D. Stride: a web server for secondary structure assignment from known atomic coordinates of proteins. **Nucleic Acids Res.**, Oxford University Press, v. 32, n. suppl 2, p. W500–W502, 2004.

HONG, Z. et al. A multi-angle hierarchical differential evolution approach for multimodal optimization problems. **IEEE Access**, IEEE, v. 8, p. 178322–178335, 2020.

HOVMÖLLER, S.; ZHOU, T.; OHLSON, T. Conformations of amino acids in proteins. **Acta Crystallogr. Sect. D-Biol. Crystallogr.**, International Union of Crystallography, v. 58, n. 5, p. 768–776, 2002.

HUANG, C.; LI, Y.; YAO, X. A survey of automatic parameter tuning methods for metaheuristics. **IEEE Trans. Evol. Comput.**, IEEE, v. 24, n. 2, p. 201–216, 2019.

HUSSAIN, K. et al. Metaheuristic research: a comprehensive survey. **Artif. Intell. Rev.**, Springer, v. 52, n. 4, p. 2191–2233, 2019.

HUTSON, M. AI protein-folding algorithms solve structures faster than ever. **Nature**, Nature Research, v. 22, p. 1–1, 2019.

INOSTROZA-PONTA, M. et al. Exploring the high selectivity of 3-D protein structures using distributed memetic algorithms. **J. Comput. Sci.**, Elsevier, v. 41, n. 101087, p. 1–17, 2020.

INOSTROZA-PONTA, M.; FARFÁN, C.; DORN, M. A memetic algorithm for protein structure prediction based on conformational preferences of aminoacid residues. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE. **Proceedings...** Madrid, Spain: ACM, 2015. p. 1403–1404.

ISLAM, M. K.; CHETTY, M. Novel memetic algorithm for protein structure prediction. In: AUSTRALASIAN JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE. **AI 2009: Advances in Artificial Intelligence**. Melbourne, Australia: Springer, 2009, (Lecture Notes in Computer Science, v. 5866). p. 412–421.

JADON, S. S. et al. Hybrid artificial bee colony algorithm with differential evolution. **Appl. Soft Comput.**, Elsevier, v. 58, p. 11–24, 2017.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM Comput. Surv.**, ACM, v. 31, n. 3, p. 264–323, 1999.

JONG, K. A. D. **An analysis of the behavior of a class of genetic adaptive systems**. Ann Arbor, MI, USA, 1975. AAI7609381, 266 p.

JORGENSEN, W. L.; TIRADO-RIVES, J. Potential energy functions for atomic-level simulations of water and organic and biomolecular systems. **Proc. Natl. Acad. Sci. USA**, National Academy of Sciences, v. 102, n. 19, p. 6665–6670, 2005.

JUMPER, J. et al. High accuracy protein structure prediction using deep learning. **Critical Assessment of Techniques for Protein Structure Prediction (Abstract Book)**, Fourteenth round, p. 22–24, 2020.

KABSCH, W.; SANDER, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. **Biopolymers**, Wiley Online Library, v. 22, n. 12, p. 2577–2637, 1983.

KANDATHIL, S. M.; GREENER, J. G.; JONES, D. T. Recent developments in deep learning applied to protein structure prediction. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 87, n. 12, p. 1179–1189, 2019.

KARABOGA, D.; AKAY, B. A comparative study of artificial bee colony algorithm. **Appl. Math. Comput.**, Elsevier, v. 214, n. 1, p. 108–132, 2009.

KARABOGA, D.; BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. **J. Global Optim.**, Springer, v. 39, n. 3, p. 459–471, 2007.

KARABOGA, D.; BASTURK, B. On the performance of artificial bee colony (abc) algorithm. **Appl. Soft Comput.**, Elsevier, v. 8, n. 1, p. 687–697, 2008.

KARABOGA, D. et al. A comprehensive survey: artificial bee colony (abc) algorithm and applications. **Artif. Intell. Rev.**, Springer, v. 42, n. 1, p. 21–57, 2014.

KARAFOTIAS, G.; HOOGENDOORN, M.; EIBEN, Á. E. Parameter control in evolutionary algorithms: Trends and challenges. **IEEE Trans. Evol. Comput.**, IEEE, v. 19, n. 2, p. 167–187, 2014.

KAUFMANN, K. W. et al. Practically useful: what the rosetta protein modeling suite can do for you. **Biochemistry**, ACS Publications, v. 49, n. 14, p. 2987–2998, 2010.

KENNEDY, J. et al. **Swarm intelligence**. 1. ed. San Francisco, USA: Morgan Kaufmann, 2001. 512 p.

KIM, D. E. et al. Sampling bottlenecks in de novo protein structure prediction. **J. Mol. Biol.**, Elsevier, v. 393, n. 1, p. 249–260, 2009.

KIM, D. E.; CHIVIAN, D.; BAKER, D. Protein structure prediction and analysis using the Robetta server. **Nucleic Acids Res.**, Oxford University Press, v. 32, n. suppl_2, p. W526–W531, 2004.

KIM, D. E. et al. One contact for every twelve residues allows robust and accurate topology-level protein structure modeling. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 82, p. 208–218, 2014.

KINCH, L. N. et al. Casp11 target classification. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 84, n. S1, p. 20–33, 2016.

KIRAN, M. S.; FINDIK, O. A directed artificial bee colony algorithm. **Appl. Soft Comput.**, Elsevier, v. 26, p. 454–462, 2015.

KOCSIS, L.; SZEPESVÁRI, C. Bandit based monte-carlo planning. In: EUROPEAN CONFERENCE ON MACHINE LEARNING. **Proceedings...** Berlin, Germany: Springer, 2006. p. 282–293.

KOLINSKI, A.; SKOLNICK, J. Reduced models of proteins and their applications. **Polymer**, Elsevier, v. 45, n. 2, p. 511–524, 2004.

KORTEMME, T.; MOROZOV, A.; BAKER, D. An orientation-dependent hydrogen bonding potential improves prediction of specificity and structure for proteins and protein–protein complexes. **J. Mol. Biol.**, Elsevier, v. 326, n. 4, p. 1239–1259, 2003.

KRAMER, O. **A brief introduction to continuous evolutionary optimization**. 1. ed. New York, USA: Springer, 2014. 94 p. (SpringerBriefs in Computational Intelligence).

KRINK, T.; VESTERSTROM, J. S.; RIGET, J. Particle swarm optimisation with spatial particle extension. In: CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Honolulu, HI, USA: IEEE, 2002. v. 2, p. 1474–1479.

KRYSHTAFOVYCH, A. et al. Critical assessment of methods of protein structure prediction (casp)—round xiii. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 87, n. 12, p. 1011–1020, 2019.

KUHLMAN, B.; BAKER, D. Native protein sequences are close to optimal for their structures. **Proc. Natl. Acad. Sci. USA**, National Academy of Sciences, v. 97, n. 19, p. 10383–10388, 2000.

KUHLMAN, B.; BRADLEY, P. Advances in protein structure prediction and design. **Nat. Rev. Mol. Cell Biol.**, Nature Research, v. 20, p. 681–697, 2019.

KVASOV, D. E.; MUKHAMETZHANOV, M. S. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. **Appl. Math. Comput.**, Elsevier, v. 318, p. 245–259, 2018.

LACERDA, M. G. P. de et al. A systematic literature review on general parameter control for evolutionary and swarm-based algorithms. **Swarm Evol. Comput.**, Elsevier, v. 60, n. 100777, p. 1–10, 2021.

LACROIX, B.; MOLINA, D.; HERRERA, F. Region-based memetic algorithm with archive for multimodal optimisation. **Inf. Sci.**, Elsevier, v. 367, p. 719–746, 2016.

LASKOWSKI, R. A.; WATSON, J. D.; THORNTON, J. M. Profunc: a server for predicting protein function from 3d structure. **Nucleic Acids Res.**, Oxford University Press, v. 33, p. 89–93, 2005.

LAZARIDIS, T.; KARPLUS, M. Effective energy functions for protein structure prediction. **Curr. Opin. Struct. Biol.**, Elsevier, v. 10, n. 2, p. 139–145, 2000.

LEAVER-FAY, A. et al. Scientific benchmarks for guiding macromolecular energy function improvement. **Methods Enzymol.**, NIH Public Access, v. 523, p. 109, 2013.

LEAVER-FAY, A. et al. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. **Methods Enzymol.**, v. 487, p. 545–574, 2011.

LESK, A. **Introduction to protein science: architecture, function, and genomics**. 2. ed. New York, USA: Oxford university press, 2010. 455 p.

LEUNG, Y.-W.; WANG, Y. An orthogonal genetic algorithm with quantization for global numerical optimization. **IEEE Trans. Evol. Comput.**, IEEE, v. 5, n. 1, p. 41–53, 2001.

LEVITT, M. et al. Protein folding: the endgame. **Annu. Rev. Biochem.**, Annual Reviews, v. 66, n. 1, p. 549–579, 1997.

LI, G. et al. Artificial bee colony algorithm with gene recombination for numerical function optimization. **Appl. Soft Comput.**, Elsevier, v. 52, p. 146–159, 2017.

LI, G.; NIU, P.; XIAO, X. Development and investigation of efficient artificial bee colony algorithm for numerical function optimization. **Appl. Soft Comput.**, Elsevier, v. 12, n. 1, p. 320–332, 2012.

LI, J.-P. et al. A species conserving genetic algorithm for multimodal function optimization. **Evol. Comput.**, MIT Press, v. 10, n. 3, p. 207–234, 2002.

LI, L.; TANG, K. History-based topological speciation for multimodal optimization. **IEEE Trans. Evol. Comput.**, IEEE, v. 19, n. 1, p. 136–150, 2014.

LI, X. Efficient differential evolution using speciation for multimodal function optimization. In: CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION. **Proceedings...** Washington DC, USA: ACM, 2005. p. 873–880.

LI, X. Niching without niching parameters: particle swarm optimization using a ring topology. **IEEE Trans. Evol. Comput.**, IEEE, v. 14, n. 1, p. 150–169, 2009.

LI, X.; ENGELBRECHT, A.; EPITROPAKIS, M. G. **Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization**. Melbourne, Australia, 2013. 10 p.

LI, X. et al. Seeking multiple solutions: an updated survey on niching methods and their applications. **IEEE Trans. Evol. Comput.**, IEEE, v. 21, n. 4, p. 518–538, 2016.

LI, Y.; YU, J.; TAKAGI, H. Niche method complementing the nearest-better clustering. In: IEEE SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE. **Proceedings...** Xiamen, China: IEEE, 2019. p. 3065–3071.

LI, Y. et al. Deducing high-accuracy protein contact-maps from a triplet of coevolutionary matrices through deep residual convolutional networks. **bioRxiv**, Cold Spring Harbor Laboratory, PP, n. PPR223248, p. 1–20, 2020.

LIANG, J. J.; QU, B. Y.; SUGANTHAN, P. N. **Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization**. Singapore, 2013. 32 p.

LIGABUE-BRAUN, R. et al. Everyone is a protagonist: Residue conformational preferences in high-resolution protein structures. **J. Comput. Biol.**, Mary Ann Liebert, Inc., v. 25, n. 4, p. 451–465, 2018.

LIU, Q. et al. Double-layer-clustering differential evolution multimodal optimization by speciation and self-adaptive strategies. **Inf. Sci.**, Elsevier, v. 545, p. 465–486, 2021.

LIU, S.-H. et al. A parameter control method of evolutionary algorithms using exploration and exploitation measures with a practical application for fitting sovova's mass transfer model. **Appl. Soft Comput.**, Elsevier, v. 13, n. 9, p. 3792–3805, 2013.

LOBANOV, M. Y.; BOGATYREVA, N.; GALZITSKAYA, O. Radius of gyration as an indicator of protein structure compactness. **J. Mol. Biol.**, Springer, v. 42, n. 4, p. 623–628, 2008.

LODISH, H. et al. **Molecular cell biology**. 6. ed. New York, USA: W.H. Freeman, 2007. 973 p.

LUKE, S. **Essentials of Metaheuristics**. 2. ed. Morrisville, North Carolina, USA: Lulu Press, Inc., 2013. 263 p. Available for free at https://cs.gmu.edu/ sean/book/metaheuristics.

MACKERREL, A. Empirical force fields. In: **Computational methods for protein structure prediction and modeling**. 1. ed. New York, USA: Springer, 2010. chp. 2, p. 45–69.

MAHFOUD, S. W. Crowding and preselection revisited. In: PARALLEL PROBLEM SOLVING FROM NATURE. **Proceedings...** Amsterdam, Netherlands: Elsevier, 1992. v. 2, p. 27–36.

MAHFOUD, S. W. **Niching methods for genetic algorithms**. Urbana, Illinois, USA, 1995. v. 51, n. 95001, 62–94 p.

MAREE, S. et al. Real-valued evolutionary multi-modal optimization driven by hill-valley clustering. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE. **Proceedings...** Kyoto, Japan: ACM, 2018. p. 857–864.

MARTÍ-RENOM, M. A. et al. Comparative protein structure modeling of genes and genomes. **Annu. Rev. Biophys. Biomol. Struct.**, Annual Reviews, v. 29, n. 1, p. 291–325, 2000.

MCREE, D. E. **Practical protein crystallography**. 2. ed. London, UK: Academic press, 1999. 477 p.

MIRNY, L.; SHAKHNOVICH, E. Protein folding theory: from lattice to all-atom models. **Annu. Rev. Biophys. Biomol. Struct.**, Annual Reviews, v. 30, n. 1, p. 361–396, 2001.

MOLINA, D.; LOZANO, M.; HERRERA, F. Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Barcelona, Spain: IEEE, 2010. p. 1–8.

MORALES-CASTAÑEDA, B. et al. A better balance in metaheuristic algorithms: Does it exist? **Swarm Evol. Comput.**, Elsevier, v. 54, n. 100671, p. 1–23, 2020.

MOSCATO, P. **On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms**. Pasadena, California, USA, 1989. C3P, n. 826, 68 p.

MOSCATO, P.; COTTA, C. A modern introduction to memetic algorithms. In: **Handbook of Metaheuristics**. 1. ed. Boston, MA: Springer, 2010. v. 146, p. 141–183.

MOSCATO, P.; COTTA, C. An accelerated introduction to memetic algorithms. In: **Handbook of Metaheuristics**. 1. ed. Cham, Switzerland: Springer, Cham, 2019. v. 272, p. 275–309.

MOULT, J. et al. Critical assessment of methods of protein structure prediction (casp)-round xii. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 86, p. 7–15, 2018.

NERI, F.; COTTA, C.; MOSCATO, P. **Handbook of memetic algorithms**. 1. ed. Heidelberg, Germany: Springer, 2012. 368 p.

NERI, F.; TIRRONEN, V. Recent advances in differential evolution: a survey and experimental analysis. **Artif. Intell. Rev.**, Springer, v. 33, n. 1-2, p. 61–106, 2010.

NEUMAIER, A. Molecular modeling of proteins and mathematical prediction of protein structure. **SIAM review**, SIAM, v. 39, n. 3, p. 407–460, 1997.

OSGUTHORPE, D. J. Ab initio protein folding. **Curr. Opin. Struct. Biol.**, Elsevier, v. 10, n. 2, p. 146–152, 2000.

OVCHINNIKOV, S. et al. Protein structure prediction using rosetta in casp12. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 86, p. 113–121, 2018.

O'MEARA, M. J. et al. Combined covalent-electrostatic model of hydrogen bonding improves structure prediction with rosetta. **J. Chem. Theory Comput.**, ACS Publications, v. 11, n. 2, p. 609–622, 2015.

PARPINELLI, R. S. et al. A review of techniques for online control of parameters in swarm intelligence and evolutionary computation algorithms. **Int. J. Bio-Inspir. Com.**, Inderscience Publishers, v. 13, n. 1, p. 1–20, 2019.

PAULING, L.; COREY, R. B. The pleated sheet, a new layer configuration of polypeptide chains. **Proc. Natl. Acad. Sci. USA**, National Academy of Sciences, v. 37, n. 5, p. 251–256, 1951.

PAULING, L.; COREY, R. B.; BRANSON, H. R. The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. **Proc. Natl. Acad. Sci. USA**, National Academy of Sciences, v. 37, n. 4, p. 205–211, 1951.

PENG, H.; DENG, C.; WU, Z. Best neighbor-guided artificial bee colony algorithm for continuous optimization problems. **Soft Comput.**, Springer, v. 23, n. 18, p. 8723–8740, 2019.

PHAN, H. D. et al. A survey of dynamic parameter setting methods for nature-inspired swarm intelligence algorithms. **Neural. Comput. Appl.**, Springer, v. 32, n. 2, p. 567–588, 2020.

POLÁKOVÁ, R.; BUJOK, P. Adaptation of population size in differential evolution algorithm: An experimental comparison. In: INTERNATIONAL CONFERENCE ON SYSTEMS, SIGNALS AND IMAGE PROCESSING. **Proceedings...** Maribor, Slovenia: IEEE, 2018. p. 1–5.

PREUSS, M. Niching the CMA-ES via nearest-better clustering. In: CONFERENCE COMPANION ON GENETIC AND EVOLUTIONARY COMPUTATION. **Proceedings...** Portland, Oregon, USA: ACM, 2010. p. 1711–1718.

PREUSS, M. Improved topological niching for real-valued global optimization. In: EUROPEAN CONFERENCE ON THE APPLICATIONS OF EVOLUTIONARY COMPUTATION. **Proceedings...** Málaga, Spain: Springer, 2012. p. 386–395.

PRICE, K. et al. **Problem definitions and evaluation criteria for the 100-digit challenge special session and competition on single objective numerical optimization**. Singapore, 2018. 21 p.

PRICE, K. et al. **The 2019 100-Digit Challenge on Real-Parameter, Single Objective Optimization: Analysis of Results**. Singapore, 2019. 22 p.

PRUITT, K. D.; TATUSOVA, T.; MAGLOTT, D. R. Ncbi reference sequence (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. **Nucleic Acids Res.**, Oxford University Press, v. 33, n. suppl 1, p. D501–D504, 2005.

QING, L. et al. Crowding clustering genetic algorithm for multimodal function optimization. **Appl. Soft Comput.**, Elsevier, v. 8, n. 1, p. 88–95, 2008.

QU, B.-Y.; SUGANTHAN, P. N.; DAS, S. A distance-based locally informed particle swarm model for multimodal optimization. **IEEE Trans. Evol. Comput.**, IEEE, v. 17, n. 3, p. 387–402, 2013.

QU, B.-Y.; SUGANTHAN, P. N.; LIANG, J.-J. Differential evolution with neighborhood mutation for multimodal optimization. **IEEE Trans. Evol. Comput**, IEEE, v. 16, n. 5, p. 601–614, 2012.

RAHNAMAYAN, S.; TIZHOOSH, H. R.; SALAMA, M. M. Opposition-based differential evolution. **IEEE Trans. Evol. Comput.**, IEEE, v. 12, n. 1, p. 64–79, 2008.

RAMACHANDRAN, G.; RAMAKRISHNAN, C.; SASISEKHARAN, V. Stereochemistry of polypeptide chain configurations. **J. Mol. Biol.**, Elsevier, v. 7, p. 95–99, 1963.

ROCHA, G. K. et al. Using crowding-distance in a multiobjective genetic algorithm for protein structure prediction. In: GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE. **Proceedings...** New York, USA: ACM, 2016. p. 1285–1292.

ROHL, C. A. et al. Protein structure prediction using rosetta. **Methods Enzymol.**, Elsevier, v. 383, p. 66–93, 2004.

SABAR, N. R.; ALETI, A. An adaptive memetic algorithm for the architecture optimisation problem. In: AUSTRALASIAN CONFERENCE ON ARTIFICIAL LIFE AND COMPUTATIONAL INTELLIGENCE. **Proceedings...** Geelong, VIC, Australia: Springer, 2017. p. 254–265.

SALEH, S.; OLSON, B.; SHEHU, A. A population-based evolutionary search approach to the multiple minima problem in de novo protein structure prediction. **BMC Struct. Biol.**, BioMed Central Ltd, v. 13, n. Suppl 1, p. S4, 2013.

SCHAARSCHMIDT, J. et al. Assessment of contact predictions in casp12: Co-evolution and deep learning coming of age. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 86, p. 51–66, 2018.

SCHEEF, E. D.; FINK, J. L. Fundamentals of protein structure. In: **Structural Bioinformatics**. 2. ed. New Jersey, USA: John Wiley & Sons, Inc., 2009. chp. 2, p. 15–40.

SENIOR, A. W. et al. Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (casp13). **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 87, n. 12, p. 1141–1148, 2019.

SENIOR, A. W. et al. Improved protein structure prediction using potentials from deep learning. **Nature**, Nature Research, v. 577, n. 7792, p. 706–710, 2020.

SER, J. D. et al. Bio-inspired computation: Where we stand and what's next. **Swarm Evol. Comput.**, Elsevier, v. 48, p. 220–250, 2019.

SERGEYEV, Y. D.; KVASOV, D.; MUKHAMETZHANOV, M. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. **Sci. Rep.**, Nature Research, v. 8, n. 1, p. 1–9, 2018.

SHEHU, A.; KAVRAKI, L. E.; CLEMENTI, C. Multiscale characterization of protein conformational ensembles. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 76, n. 4, p. 837–851, 2009.

SHRESTHA, R. et al. Assessing the accuracy of contact predictions in casp13. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 87, n. 12, p. 1058–1068, 2019.

SIMONS, K. T. et al. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. **J. Mol. Biol.**, Elsevier, v. 268, n. 1, p. 209–225, 1997.

SOLIS, F.; WETS, R.-B. Minimization by random search techniques. **Math. Oper. Res.**, Informs, v. 6, n. 1, p. 19–30, 1981.

SONG, Y. et al. High-resolution comparative modeling with rosettacm. **Structure**, Elsevier, v. 21, n. 10, p. 1735–1742, 2013.

SRINIVAS, M.; PATNAIK, L. M. Adaptive probabilities of crossover and mutation in genetic algorithms. **IEEE Trans. Syst. Man Cybern. Syst.**, IEEE, v. 24, n. 4, p. 656–667, 1994.

STILLINGER, F. H.; HEAD-GORDON, T.; HIRSHFELD, C. L. Toy model for protein folding. **Phys. Rev. E**, APS, v. 48, n. 2, p. 1469–1477, 1993.

STORN, R.; PRICE, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. **J. Global Optim.**, Springer, v. 11, n. 4, p. 341–359, 1997.

SUGANTHAN, P. N. et al. **Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization**. Singapore, 2005. 50 p.

SYSWERDA, G. Uniform Crossover in Genetic Algorithms. In: INTERNATIONAL CONFERENCE ON GENETIC ALGORITHMS. **Proceedings...** San Mateo, California: Morgan Kaufmann Publishers, Inc., 1989. p. 2–9.

TALBI, E.-G. Common concepts for metaheuristics. In: **Metaheuristics: from design to implementation**. Weinheim, Germany: John Wiley & Sons, Inc., 2009. v. 74, chp. 1, p. 1–86.

TANABE, R.; FUKUNAGA, A. Success-history based parameter adaptation for differential evolution. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Cancun, Mexico: IEEE, 2013. p. 71–78.

TANABE, R.; FUKUNAGA, A. S. Improving the search performance of shade using linear population size reduction. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Beijing, China: IEEE, 2014. p. 1658–1665.

TANG, K. et al. Multi-strategy adaptive particle swarm optimization for numerical optimization. **Eng. Appl. Artif. Intell.**, Elsevier, v. 37, p. 9–19, 2015.

TEZEL, B. T.; MERT, A. A cooperative system for metaheuristic algorithms. **Expert Syst. Appl.**, Elsevier, v. 65, n. 113976, p. 1–15, 2020.

THOMSEN, R. Multimodal optimization using crowding-based differential evolution. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Portland, OR, USA: IEEE, 2004. v. 2, p. 1382–1389.

TOFFOLO, A.; BENINI, E. Genetic diversity as an objective in multi-objective evolutionary algorithms. **Evol. Comput.**, MIT Press, v. 11, n. 2, p. 151–167, 2003.

TOMPA, P. Intrinsically unstructured proteins. **Trends Biochem. Sci.**, Elsevier, v. 27, n. 10, p. 527–533, 2002.

TORRES-JIMÉNEZ, J.; PAVÓN, J. Applications of metaheuristics in real-life problems. **Prog. Artif. Intell.**, Springer, v. 2, p. 175–176, 2014.

TZANETOS, A.; DOUNIAS, G. Nature inspired optimization algorithms or simply variations of metaheuristics? **Artif. Intell. Rev.**, Springer, v. 54, p. 1841–1862, 2020.

UNGER, R.; MOULT, J. Finding the lowest free energy conformation of a protein is an np-hard problem: proof and implications. **Bull. Math. Biol.**, Springer, v. 55, n. 6, p. 1183–1198, 1993.

UNWIN, P. N. T.; HENDERSON, R. Molecular structure determination by electron microscopy of unstained crystalline specimens. **J. Mol. Biol.**, Elsevier, v. 94, n. 3, p. 425IN13433–432IN18440, 1975.

URSEM, R. K. Multinational gas: Multimodal optimization techniques in dynamic environments. In: CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION. **Proceedings...** San Francisco, CA, USA: Morgan Kaufmann Publishers, Inc., 2000. p. 19–26.

WANG, X. et al. A multilevel sampling strategy based memetic differential evolution for multimodal optimization. **Neurocomputing**, Elsevier, v. 334, p. 79–88, 2019.

WANG, Z.-J. et al. Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems. **IEEE Trans. Evol. Comput.**, IEEE, v. 22, n. 6, p. 894–908, 2017.

WANG, Z.-J. et al. Automatic niching differential evolution with contour prediction approach for multimodal optimization problems. **IEEE Trans. Evol. Comput.**, IEEE, v. 24, n. 1, p. 114–128, 2019.

WANG, Z.-J.; ZHAN, Z.-H.; ZHANG, J. Distributed minimum spanning tree differential evolution for multimodal optimization problems. **Soft Comput.**, Springer, v. 23, n. 24, p. 13339–13349, 2019.

WHISSTOCK, J. C.; LESK, A. M. Prediction of protein function from protein sequence and structure. **Q. Rev. Biophys.**, Cambridge University Press, v. 36, n. 3, p. 307–340, 2003.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. **IEEE Trans. Evol. Comput**, IEEE, v. 1, n. 1, p. 67–82, 1997.

WONG, W.; MING, C. I. A review on metaheuristic algorithms: Recent trends, benchmarking and applications. In: INTERNATIONAL CONFERENCE ON SMART COMPUTING & COMMUNICATIONS. **Proceedings...** Sarawak, Malaysia: IEEE, 2019. p. 1–5.

XU, J.; WANG, S. Analysis of distance-based protein structure prediction by deep learning in casp13. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 87, n. 12, p. 1069–1081, 2019.

XU, J.; ZHANG, J. Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis. In: CHINESE CONTROL CONFERENCE. **Proceedings...** Nanjing, China: IEEE, 2014. p. 8633–8638.

XUE, Y. et al. A self-adaptive artificial bee colony algorithm based on global best for global optimization. **Soft Comput.**, Springer, v. 22, n. 9, p. 2935–2952, 2018.

YANG, Q. et al. Multimodal estimation of distribution algorithms. **IEEE Trans. Cybern.**, IEEE, v. 47, n. 3, p. 636–650, 2016.

YANG, Q. et al. Adaptive multimodal continuous ant colony optimization. **IEEE Trans. Evol. Comput.**, IEEE, v. 21, n. 2, p. 191–205, 2016.

YAO, J.; KHARMA, N.; GROGONO, P. Bi-objective multipopulation genetic algorithm for multimodal function optimization. **IEEE Trans. Evol. Comput.**, IEEE, v. 14, n. 1, p. 80–102, 2009.

YURIEV, E.; HOLIEN, J.; RAMSLAND, P. A. Improvements, trends, and new ideas in molecular docking: 2012–2013 in review. **J. Mol. Recognit.**, Wiley Online Library, v. 28, n. 10, p. 581–604, 2015.

ZEMLA, A. Lga: a method for finding 3d similarities in protein structures. **Nucleic Acids Res.**, Oxford University Press, v. 31, n. 13, p. 3370–3374, 2003.

ZHANG, J.; SANDERSON, A. C. Jade: adaptive differential evolution with optional external archive. **IEEE Trans. Evol. Comput.**, IEEE, v. 13, n. 5, p. 945–958, 2009.

ZHANG, Y.; SKOLNICK, J. Scoring function for automated assessment of protein structure template quality. **Proteins: Struct. Funct. Bioinf.**, Wiley Online Library, v. 57, n. 4, p. 702–710, 2004.

ZHANG, Y.-H. et al. A tree-structured random walking swarm optimizer for multimodal optimization. **Appl. Soft Comput.**, v. 78, p. 94–108, 2019.

ZHAO, H.; ZHAN, Z.-H.; ZHANG, J. Adaptive guidance-based differential evolution with iterative feedback archive strategy for multimodal optimization problems. In: IEEE CONGRESS ON EVOLUTIONARY COMPUTATION. **Proceedings...** Glasgow, UK: IEEE, 2020. p. 1–8.

ZHU, G.; KWONG, S. Gbest-guided artificial bee colony algorithm for numerical function optimization. **Appl. Math. Comput.**, Elsevier, v. 217, n. 7, p. 3166–3173, 2010.