University of Pernambuco
Polytechnic School of Pernambuco
Postgraduate Program in Computing Engineering

Lumadaiara do Nascimento Vitorino

An Operator based on Artificial Bees to Generate Diversity for Particle Swarm Optimization

Master's Dissertation

Recife, July 2012

University de Pernambuco
Polytechnic School of Pernambuco
Postgraduate Program in Computing Engineering

Lumadaiara do Nascimento Vitorino

An Operator based on Artificial Bees to Generate Diversity for Particle Swarm Optimization

Master's Dissertation

Dissertation presented to Postgraduate Program in Computing Engineering as a partial requirement for obtaining the Degree of Master of Science in Computer Engineering.

Supervisor: Prof. DSc. Carmelo José Albanez Bastos Filho

Recife, July de 2012

DEDICATION

Firstly, I want to thank God for giving me health and peace, and to make everything possible. I have to thank a lot to my father for his understanding and to my mother for her unconditional support. A special thanks to my brother by his cheer, support and for motivating me during this journey.

I am very grateful to my supervisor, Professor Carmelo, for all learning and guidance. When working by his side, I found my vocation as a professor. He is my example of professor, researcher and advisor. What he taught me is not found in books. He taught me to believe in the potential of students, motivate them, let them excited during classes and respect people. Thus, he won the admiration of all the people. I thank all the professors of the Postgraduate Program in Computing Engineering for their contribution to my development. I also thank the undergraduate student Sergio Ribeiro due to our successful partnership.

I want to thank my friends of now and forever, Cesar, Caio, Thiago, George, Victor and Silvia for all the support. Despite the distance, they always were with me. It is always good to make new friends. Finally, I want to thank all the new friends and colleagues that I did during these two years, Paulo Roger, Debora, Marcelo, Renatha, Rodrigo (teoria) because my master's degree would not be the same without their support.

A special thank to the secretaries Ana Georgina and Julia by their conversations and help. They welcomed me with a smile when I arrived and said "Hi" always.

I know that I may lose contact with most of these people due the circumstances, but I want to state my thanks for they having entered into my life.

EPIGRAPH

"Everyone here has the sense that right now is one of those moments when we are influencing the future." Steve Jobs

Recife, July 2012

PORTUGUESE LANGUAGE ABSTRACT

A área de Inteligência de enxames se tornou uma das áreas de pesquisa da inteligência computacional que mais se expandiu nos últimos anos. Os algoritmos de inteligência de enxames são geralmente usados para otimização e busca, e normalmente apresentam mecanismos para simultaneamente representar sucesso, garantir convergência e manter a diversidade. O algoritmo otimização por enxames de partículas (PSO) utiliza a melhor posição encontrada no espaço de busca e intensifica a busca em determinadas regiões mais promissoras. Por outro lado, o algoritmo de otimização por colmeias artificiais (ABC) utiliza o conceito de fontes de alimento, as quais as abelhas exploram e caso não hajam melhorias de qualidade, elas vão buscar novas fontes de alimento.

Estes algoritmos apresentam excelentes habilidades para resolver problemas complexos, contudo muitos perdem a sua eficiência quanto aplicados em problemas de alta dimensionalidade, por exemplo com 1000 variáveis de decisão. Diante deste cenário, combinar algoritmos de inteligência de enxames pode ser uma alternativa para resolver problemas complexos de alta dimensionalidade.

Nesta dissertação, a proposta é combinar o algoritmo PSO com comportamento adaptativo, que possui um bom mecanismo de convergência, com o ABC, com a sua capacidade de manter a diversidade. Esse algoritmo combinado é chamado de *Adaptive Bee and Particle Swarm Optimization* (ABeePSO). O desempenho do algoritmo ABeePSO foi avaliado utilizando as funções de teste apresentadas no (*IEEE Congress on Evolutionary Computation 2010*). Foram analisadas convergência, diversidade e escalabilidade. Também foram realizadas comparações com outras técnicas de inteligência de enxames presentes na literatura. Os resultados indicam que o desempenho do ABeePSO foi superior principalmente em espaços de busca de alta dimensionalidade.

Recife, July 2012

ENGLISH LANGUAGE ABSTRACT

Swarm Intelligence is one of the research areas of computational intelligence that most evolved in the last years. The swarm intelligence algorithms are bio-inspired and are usually used for search and optimization problems. They often have mechanisms to simultaneously represent success, guarantee of convergence and maintain diversity. Particle Swarm Optimization (PSO) is one of the most known swarm intelligence algorithms. PSO uses the best positions found in the search space and intensifies the search in some of the most promising regions. On the other hand, the Artificial Bee Colony (ABC) algorithm uses the concept of food sources. The bees explore these food source, and if the quality of these food sources are not improving over the iterations, these sources are abandoned.

These algorithms have excellent skills to solve complex problems. However, many of them lose their efficiency when applied to optimization problems with high dimensionality, *e.g.* 1000 decision variables. Thus, a proper combination of swarm intelligence algorithms that present different features can be a suitable alternative to solve optimization problems with high dimensionality.

In this dissertation, the proposal is to combine the PSO algorithm with adaptive behavior (Adaptive PSO - APSO) with the ABC. The APSO algorithm has a good mechanism to accelerate the convergence and ABC has the ability to maintain diversity. This combination is called *Bee and Adaptive Particle Swarm Optimization* (*ABeePSO*). The performance of the ABeePSO algorithm was evaluated in the benchmark optimization suite proposed in (*IEEE Congress on Evolutionary Computation 2010*). We performed a detailed analysis regarding convergence, diversity and scalability. We also compare our approach to other well known swarm intelligence algorithms. The results suggest that the performance of the ABeePSO algorithm is superior, mainly in high dimensionality search spaces.

Recife, July 2012

Contents

List of Figures **IX**

List of Tables **XI**

List of Symbols **XII**

List of Algorithms **XIV**

1 Introduction **10**

1.1 Motivation and Objectives 12

1.2 Contributions 13

1.3 Document Structure 13

2 Swarm Intelligence Fundamentals **15**

2.1 Particle Swarm Optimization - PSO 15

2.1.1 Pseudocode of the PSO Algorithm 17

2.2 Adaptive Particle Swarm Optimization - APSO 17

2.2.1 Estimation of Evolutionary Factor 17

2.2.2 Classification of Evolutionary State 19

2.2.3 Adaptation of Acceleration Coefficients 20

2.2.4 Elitist Learning Strategy 21

2.2.5 Adaptation of Inertial Factor 22

2.2.6 Pseudocode of the APSO Algorithm 22

2.3 Clan Particle Swarm Optimization - ClanPSO 22

2.3.1 Delegation of Leaders 24

2.3.2 Conference of Leaders 25

2.3.3 The Information shared within the clans 25

2.3.4 Pseudocode of the ClanPSO Algorithm 25

2.4 Clan Adaptive Particle Swarm Optimization - ClanAPSO 26

2.4.1 Pseudocode of the ClanAPSO Algorithm 27

2.5 Artificial Bee Colony - ABC 27

2.5.1 Quality of Food Source 28

2.5.2 Update of Food Source 28

2.5.3 Choose of Food Source by Onlookers Bees 29

2.5.4 Stagnation of Food Source 29

2.5.5 Pseudocode of the ABC Algorithm 29
2.6 Fish School Search - FSS 29
2.6.1 Individual Movement Operator 31
2.6.2 Feeding Operator 31
2.6.3 Collective-Instinctive Movement Operator 32
2.6.4 Collective-Volitive Movement Operator 32
2.6.5 Pseudocode of the FSS Algorithm 33
2.7 Discussion about the Swarm Intelligence Algorithms 33

3 Our Proposal: An Operator based on Artificial Bees to Generate Diversity for Particle Swarm Optimization 36
3.1 Motivation 36
3.2 Adaptive Bee and Particle Swarm Optimization 37
3.2.1 Guide Bees Movement 37
3.2.2 Guided Bees Movement 38
3.2.3 Classification of Evolutionary Factor 38
3.2.4 Stagnation Counter 40
3.2.5 Pseudocode of the ABeePSO Algorithm 40
3.3 Clan Adaptive Bee and Particle Swarm Optimization 40
3.3.1 Pseudocode of the ClanABeePSO Algorithm 41

4 Methodology and Experiments Setup 44
4.1 Benchmark Functions 44
4.1.1 Separable Functions 48
4.1.2 Single-group m-nonseparable Functions 49
4.1.3 $D/2m$-group m-nonseparable Functions 50
4.1.4 D/m-group m-nonseparable Functions 51
4.1.5 Nonseparable Functions 52
4.1.6 Definition of Search Space Boundaries and Global Optimum 52
4.2 Experimental Methodology 53
4.2.1 Environment of the Simulations 53
4.2.2 Metrics of Evaluation 54
4.2.3 Experiments Setup 54

5 Simulation Results 56
5.1 Parametric Analysis of the ABeePSO 56
5.1.1 Analysis of Dispersion Step 56
5.1.2 Analysis of Acceleration Coefficients (c_1 and c_2) 57
5.1.3 Analysis of Intervals of Fuzzy Membership Functions 57
5.1.4 Analysis of Stagnation Counter 59
5.2 Performance comparison among ABeePSO, PSO, APSO and ABC . 59
5.2.1 Analysis of Convergence 59
5.2.2 Analysis of Scalability 59
5.3 Initial Analysis of the ClanABeePSO Algorithm 62

6 Final Considerations and Future Works **74**
6.1 Conclusions and Discussions of the Results 74
6.2 Future Works 75

List of Figures

1.1 Example of a bird flock in the nature (Adapted from: [Sig12]). 11
1.2 Example of bee colony in the nature (Adapted from: [Eco12]). 11
1.3 Example of a fish school in the nature (Adapted from: [Cam12]). 12

2.1 Standard PSO communication topologies: (a) Star Topology used in g_{best}, (b) Ring Topology used in l_{best} and (c) Von Neumann Topology. 16
2.2 Fuzzy membership functions for the APSO algorithm. 19
2.3 Example of division of the population in sub-swarms. 24
2.4 Example of definition of clans leaders (A, B, C and D). 24
2.5 Types of conference in the ClanPSO: (a) global conference and (b) local conference. 25

3.1 Graphic illustration of logistic function of Dispersion step $step_d$. 38
3.2 Fuzzy membership functions for the ABeePSO algorithm. 39

4.1 Illustration of Sphere function: (a) Surface and (b) Contour. 45
4.2 Illustration of Elliptic function: (a) Surface and (b) Contour. 45
4.3 Illustration of Schwefel's Problem 1.2: (a) Surface and (b) Contour. 46
4.4 Illustration of Rosenbrock's function: (a) Surface and (b) Contour. . 47
4.5 Illustration of Rastrigin's function: (a) Surface and (b) Contour. . . 47
4.6 Illustration of Ackley's function: (a) Surface and (b) Contour. . . . 48
4.7 Illustration of the DOSA main screen. 54

5.1 Performance of the ABeePSO algorithm as a function of α for (a) $F1$ and (b) $F16$, considering β = 0.1 (squares), β = 0.2 (circles) and β = 0.5 (triangle). 57
5.2 Different configurations to analysis of intervals of fuzzy membership functions: (a) Configuration 2 and (b) Configuration 3. 58
5.3 Analysis of Convergence: Fitness function evolution of the ABC, APSO, PSO and ABeePSO algorithms for the D = 100 in (a) $F3$, (b) $F5$, (c) $F12$ and (d) $F14$ functions. 60
5.4 Analysis of the Dimensionality: Boxplot of the fitness for the $F7$ function varying the number of dimensions for (a) D = 100, (b) D = 300, (c) D = 500 and (d) D = 1,000. 61

5.5 Analysis of the Dimensionality: Boxplot of the fitness for the $F11$ function varying the number of dimensions for (a) $D = 100$, (b) $D = 300$, (c) $D = 500$ and (d) $D = 1{,}000$. 61

List of Tables

2.1 Strategies for the control of acceleration coefficients c_1 and c_2. 20
2.2 The number of parameters of each algorithm. 35

4.1 Benchmark functions, search space boundaries and global optimum position. 53
4.2 The number of parameters of each algorithm. 55

5.1 Impact of variation of acceleration rate in quality of search. 58
5.2 Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 100 dimensions. 63
5.3 Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 300 dimensions. 64
5.4 Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 500 dimensions. 65
5.5 Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 1,000 dimensions. 66
5.6 Results of Wilcoxon test for the comparison of our proposal to the other approaches for $100D$ and $1,000$ iterations. 67
5.7 Results of Wilcoxon test for the comparison of our proposal to the other approaches for $300D$ and $3,000$ iterations. 68
5.8 Results of Wilcoxon test for the comparison of our proposal to the other approaches for $500D$ and $5,000$ iterations. 69
5.9 Results of Wilcoxon test for the comparison of our proposal to the other approaches for $1000D$ and $10,000$ iterations. 70
5.10 Comparison of ClanABeePSO, ClanAPSO and ClanPSO algorithms with conference of leaders executing a Local PSO. 70
5.11 Comparison of ClanABeePSO, ClanAPSO and ClanPSO algorithms with conference of leaders executing a global PSO. 71

5.12 Performance comparison in terms of average fitness value and (standard deviation) between ClanABeePSO and ClaAPSO with execution of PSO in the conference of leaders for all the 20 benchmark functions for 100 dimensions. 72
5.13 Performance comparison in terms of average fitness value and (standard deviation) between ClanABeePSO and ClaAPSO with execution of APSO in the conference of leaders for all the 20 benchmark functions for 100 dimensions. 73

List of Symbols

α: The sigmoid parameter in the ABeePSO algorithm 37

β: The displacement of the point of the sigmoid function in the ABeePSO algorithm 37

$\vec{B}(t)$: The centroid of the fish school in the FSS algorithm 32

c_1: The cognitive acceleration coefficient in the PSO algorithm 16

c_2: The social acceleration coefficient in the PSO algorithm 16

δ: The acceleration rate in the APSO algorithm 21

Δf_i: The difference of fitness values between consecutive iterations . 31

f_{evol}: The evolutionary factor of the APSO algorithm 18

f_{s_i}: The food source selected by guided bee in the ABeePSO algorithm 38

$MaxTrial$: Stagnation Counter of the ABC algorithm 29

$\vec{n}(t)$: The social memory in the PSO algorithm 16

ω: The inertial factor of the in the PSO algorithm 17

$\vec{p}(t)$: The cognitive memory in the PSO algorithm 16

p_i: The probability to select the food source in the ABC algorithm . 29

σ: The standard deviation in the APSO algorithm 22

$step$: The step of guide bees in the ABeePSO algorithm 37

$step_{ind}$: The individual step in the FSS algorithm 31

$step_{vol}$: The volitive step in the FSS algorithm 33

$\vec{v}(t)$: Velocity of particle in the PSO algorithm 16

$W_i(t)$: The weight of the fish in the FSS algorithm 31

$\vec{x}(t)$: Position of particle in the PSO algorithm 16

List of Algorithms

Algorithm 1: Pseudocode of the PSO algorithm 18

Algorithm 2: Pseudocode of the APSO algorithm 23

Algorithm 3: Pseudocode of the ClanPSO algorithm 26

Algorithm 4: Pseudocode of the ClanAPSO algorithm 27

Algorithm 5: Pseudocode of the ABC algorithm 30

Algorithm 6: Pseudocode of the FSS algorithm 34

Algorithm 7: Pseudocode of the ABeePSO algorithm 42

Algorithm 8: Pseudocode of the ClanABeePSO algorithm 43

Chapter 1

Introduction

Computational Intelligence is a research area in which nature-inspired computational methodologies and approaches are proposed to tackle complex real-world problems. Computational Intelligence are often used in application where the traditional methodologies are ineffective or unfeasible. This area presents two main subareas: Fuzzy Logic [Zim01], population based techniques and artificial neural networks [RM87].

The main population based approaches are: evolutionary computation (Genetic Algorithm (GA) [Mit98] [GWH10], Differential Evolution (DE) [GyMg10] [PSL05], Evolutionary Strategies (ES) [Bey01] [BS02]), swarm based optimization methods (Particle Swarm Optimization (PSO) [EK95b] [EK95a], Ant Colony Optimization (ACO) [DC99] [DB05], Artificial Bee Colony (ABC) [Kar05] [BK06], Fish School Search (FSS) [BFLNL$^+$08], Fish Swarm Algorithm (FSA) [YATNM12], Bacterial Foraging Algorithm (BFA) [MMAK02], Bacterial Foraging Optimization (BFO) [DBDA09], Glowworm Swarm Optimization (GSO) [KG09], Firefly Algorithm (FA) [Yan09]), among others.

Swarm intelligence became one branch of the computational intelligence which experienced a huge expansion in recent years. The expression 'Swarm Intelligence' was introduced by Gerardo Beni and Jing Wang in 1989 [BW89]. The term 'swarm' is used basically to refer any structured collection of agents able to interact. The swarm intelligence systems are composed by agents that interact among each other and their environment. From this interaction, global behaviors that can not observed in the agents individually, emerge.

The swarm Intelligence techniques were originated from the study of collective behavior of microorganisms, animals and insects in the nature. They are commonly used to solve optimization and search problems. These algorithms are self-organized, distributed, autonomous, flexible and dynamic.

The main properties of a swarm intelligence system are [Ser09]: *proximity* - the agents should be able to interact; *quality* - the agents should be able to evaluate their behaviors; *diversity* - allows the system to react to unexpected situations; *stability* - not every environment variations should affect the behavior of an agent; *adaptability* - adequacy to environment variations capability.

Among several approaches, some of them are used to tackle continuous problems without constraints. We can cite some of them: PSO, ABC and FSS.

The PSO algorithm was inspired in the behavior of bird flocks (observe Figure 1.1). Basically, there is a swarm of particles, where each particle is a potential solution of the problem. The particle moves through the search space by updating its velocity and position. The velocity is calculated considering the best position found by the individual (cognitive memory) and the best position found by its neighborhood (social memory).

Figure 1.1: Example of a bird flock in the nature (Adapted from: [Sig12]).

The ABC algorithm was inspired in the behavior of honey bees (observe Figure 1.2). In this algorithm, the bee colony has three type of bees: employed bees, onlookers and scouts. The food sources are the potential solutions of the problem. Each food source has an associated employed bee. The employed bees explore the food source and share the information (through the dance of the bees) to the onlooker bee. These bees are responsible to choose a food source (preferentially, the richer one) and explore it. When the employed bee abandons the food source, it becomes a scout bee and, then, will seek a new food source.

Figure 1.2: Example of bee colony in the nature (Adapted from: [Eco12]).

The FSS algorithm was inspired in the behavior of school of fish (observe Figure 1.3). In this algorithm, the position of the fish is a potential solution of the problem. Their weights represent the success during the search process. The execution of the algorithm is based on the operators that were inspired in the individual and collective behavior of fish in order to guarantee the survivability of the entire group. As a consequence, the fish move to the most promising food source. The fish school will direct the sense of its search based on the increasement of the weight of the fish. Depending of the granularity of the search, the swarm can expand or contract.

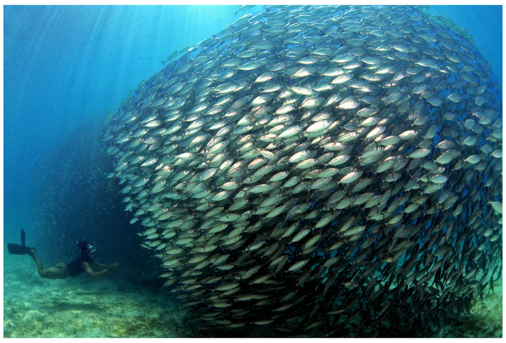


Figure 1.3: Example of a fish school in the nature (Adapted from: [Cam12]).

Each one of these algorithms has particularities, strengths and weaknesses. Moreover, they have mechanisms to simultaneously represent success, to guarantee convergence and maintain diversity. It is important to emphasize that these algorithms do not have mutually exclusive characteristics. For example, the PSO algorithm has a good convergence mechanism, but it does not have the capability to maintain the diversity of the swarm. On the other hand, the ABC algorithm has a satisfactory ability to generate diversity. Thus, we believe it is possible to combine these algorithms to provide both features.

1.1 Motivation and Objectives

The optimization problems present several challenges, such as: the "Curse of Dimensionality" effect [HKK$^+$10] [Pow07] [RNI10] and the "No Free Lunch" theorem [WM95] [WM97] [DJW02].

The curse of dimensionality refers to the phenomenon that arises when analyzing high-dimensionality search spaces (often with hundreds or thousands of dimensions). This phenomenon generally does not occur in low-dimensionality. This means that the volume of the search space increases when the dimensionality increases. In this scenario, it is expected that some optimization algorithms work well to solve problems with low dimensionality, but they can present a poor performance and high computational cost as the number of dimensions increase.

The theorem *No Free Lunch* was introduced by David Wolpert and William G. Macready [WM95]. This theorem states that if an optimization algorithm achieves better results in some classes of problems, it must present worse results in other ones. Thus, it is important to define which group of problems an optimization algorithm can tackle. In principle, there is not a single optimization algorithm that is able to solve all types of problems with the best possible performance.

The swarm intelligence algorithms have the ability to solve complex problems. Many of them have their performance mitigated when they are applied to very complex problems. Scenarios with high dimensionality, continuous variables, multimodal search spaces and variables with different degrees of dependency allows one to simulate optimization problems with characteristics that are similar to the ones presented in the real world.

In this scenario, combining swarm intelligence algorithms can be an alternative to solve complex problems with high dimensionality. Therefore, the main objective is to study the swarm intelligence approaches in order to find the main characteristics of them, indicating possible combinations so the limitations of the algorithms can be mitigated. Finally, the combined approach must be applicable to handle optimization problems with high-dimensionality.

This dissertation proposes to combine the Adaptive Particle Swarm Optimization (APSO) algorithm with the ABC algorithm. The APSO algorithm has the adaptive behavior that guarantees the convergence ability. The Artificial Bee Colony (ABC) algorithm has the diversity capability. Considering these characteristics, the proposal is to develop an operator based on artificial bees to generate diversity in order to be included it in the APSO algorithm.

1.2 Contributions

The main contribution of this dissertation is the combined algorithm called Adaptive Bee and Particle Swarm Optimization (ABeePSO). This algorithm is a proper combination of two well known swarm intelligence algorithms: Adaptive Particle Swarm Optimization (APSO) [ZZLC09] and Artificial Bee Colony (ABC) [Kar05]. The proposed algorithm has an new operator of maintain the diversity in the swarm. The proposal combines the peculiar behavior of the guide bees of the ABC algorithm and uses the centroid concept to spread the swarm, which is based on the concept proposed in the FSS algorithm. The results show that our proposal achieved better results in complex functions with high-dimensional search spaces. This means that our proposal can maintain good convergence and has the capability to escape from local minima when the search space has a high number of dimensions, mainly in multimodal functions.

1.3 Document Structure

For a better understanding of this dissertation, we indicate a sequential reading. The rest of the dissertation is organized as follows:

- Chapter 1 presented the motivation and the objectives of this dissertation;
- Chapter 2 describes the state-of-art regarding swarm intelligence optimizers;

- Chapter 3 describes our proposal, the ABeePSO algorithm;
- Chapter 4 presents the methodology and the simulation setup;
- Chapter 5 presents the results;
- Chapter 6 describes presents some conclusions and future works;
- Appendix A presents the publications of the research.

Chapter 2

Swarm Intelligence Fundamentals

Swarm intelligence algorithms were inspired by several examples provided by nature. This inspiration was found on the behavior of social microorganisms, animals and insects, such as: ant colonies [DC99][DB05][Dor97] [KK12], fish schools [BFLNL$^+$08][YATNM12], bee colonies [Kar05][TD05][Teo09][AMZ09][AMZ10][PGK$^+$05] [POA$^+$07], flocks of birds [EK95b], swarms of fireflies [KG09][Yan09], bacterial colonies [DBDA09][MMAK02], among others.

The swarm intelligence algorithms are often applied to solve optimization problems without constraints. Some of these algorithms present good abilities do tackle some types of situations that may happen during optimization processes.

In the following sections, we describe some of the most used swarm intelligence algorithms for optimization and their variations to tackle continuous search spaces. Section 2.1 presents the Particle Swarm Optimization (PSO) algorithm. Section 2.2 describes the PSO algorithm with an adaptive behavior, called Adaptive Particle Swarm Optimization (APSO). Sections 2.3 and 2.4 detail the PSO and APSO with communication topology based on clans, called Clan Particle Swarm Optimization (ClanPSO) and Clan Adaptive Particle Swarm Optimization (ClanAPSO), respectively. Section 2.5 presents the main concepts regarding Artificial Bee Colony (ABC). Finally, Section 2.6 presents the Fish School Search (FSS) algorithm.

2.1 Particle Swarm Optimization - PSO

Particle Swarm Optimization (PSO) is a computational intelligence technique proposed by James Kennedy and Russell Eberhart in 1995 [KES01] [EK95b][EK95a] [SLS12]. PSO is a population based algorithm inspired by the social behavior of flocks of birds that can be used to solve optimization and search problems. PSO was applied in many several real-world problems [WSZ$^+$04] [LFLW02] [LJG06].

In the PSO approach, the swarm is composed by a population of particles (N), where each particle i has four attributes: the position within the search space $\vec{x}_i(t)$, that represents a possible solution for the problem; the velocity of the particle $\vec{v}_i(t)$, that is used to update the position of the particle i; the best position found by the particle during the search process $\vec{p}_i(t)$(also known as the cognitive memory); and the best position found by the neighborhood of the particle $\vec{n}_i(t)$ (also known as the social memory).

The neighborhood of the particle i is the set of particles of the swarm from which the particle i is able to acquire information. This information is used to update the swarm status (*i.e.* velocity and position). The neighborhood is often defined by the communication topology. Different topologies were already proposed [KM02][KM06] and the most used topologies are depicted in Figure 2.1.

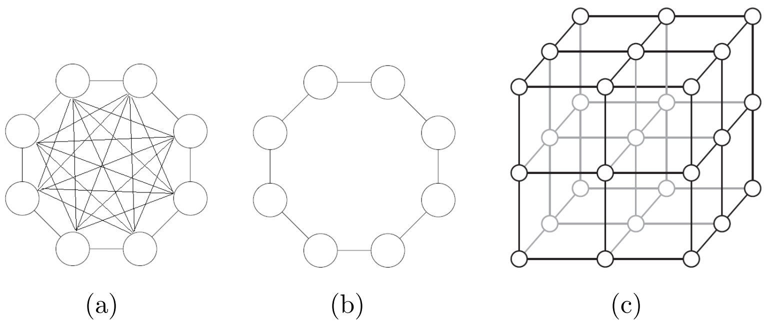


Figure 2.1: Standard PSO communication topologies: (a) Star Topology used in g_{best}, (b) Ring Topology used in l_{best} and (c) Von Neumann Topology.

Figure 2.1(a) presents the *star* topology, in which the particles share information globally through a fully-connected structure. In this case, the information is quickly spread within the swarm and allows a quick convergence of the swarm. This topology is also known as global topology or g_{best}. Figure 2.1(b) presents the *ring* topology, in which each particle is connected solely to n neighbors. In this case, the information exchange mechanism allows a slower dissemination of information and helps to avoid local minima. The Ring topology is also known as local topology or l_{best}. The Von Neumann topology is depicted in Figure 2.1(c). This is a balanced solution that consists in particles connected by a grid [BFCFdM09].

The PSO execution occurs balancing the social learning and the individual learning of the individuals within the swarm. During each algorithm iteration, the particles move through the search space by updating their velocities and positions. There are several equations that can be used to update the velocity of the particles [CK02]. The most used equations to update the velocity and the position of particles are presented in the equations (2.1) and (2.2), respectively.

$$\vec{v}(t+1) = \omega\vec{v}(t) + r_1c_1[\vec{p}(t) - \vec{x}(t)] + r_2c_2[\vec{n}(t) - \vec{x}(t)], \tag{2.1}$$

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t), \tag{2.2}$$

in which $\vec{v}(t)$ is the velocity of particle in time step t, r_1 and r_2 are random numbers selected by using an uniform probability density distribution in the interval $[0, 1]$. c_1 is the cognitive acceleration coefficient, c_2 is the social acceleration coefficient and ω is the inertial factor. $r_1c_1[\vec{p}(t) - \vec{x}(t)]$ is the cognitive component, which attracts the particle to its own best position ($\vec{p}(t)$), and $r_2c_2[\vec{n}(t) - \vec{x}(t)]$ is the social component, which attracts the particle to the best position found by its own neighborhood ($\vec{n}(t)$).

In the beginning of the algorithm execution, position ($\vec{x}(t)$) and velocity ($\vec{v}(t)$) of particles are attributed randomly. In the state-of-art [SE98] [BK07], the standard values of parameters c_1, c_2 are equal (2.05) and the inertial factor(ω) linearly decreasing from $\omega_{max} = 0.9$ to $\omega_{min} = 0.4$, along the iterations. Equation (2.3) calculates the linear decrement of the inertial factor [SE98].

$$\omega = \omega_{max} - \Big[(\omega_{max} - \omega_{min})\frac{g(t)}{g_{end}}\Big], \tag{2.3}$$

where $g(t)$ is current iteration and g_{end} is total number of iterations. The inertial factor optimizes the exploration-exploitation tradeoff. This decrement is done to allow the swarm to have a higher exploration ability in the initial iterations, and a higher exploitation capability in final iterations, when probably the swarm has found a good region of the search space to exploit.

2.1.1 Pseudocode of the PSO Algorithm

The pseudocode of the PSO algorithm is shown in Algorithm 1. One can observe that the PSO algorithm, basically, consists on particle movement in search space (lines 6 and 7) and the update process of the cognitive (lines 9 and 10) and social memories (lines 12 and 13, respectively). The adaptation of inertial factor (line 16) allows the algorithm to regulate the exploitation-exploration tradeoff.

2.2 Adaptive Particle Swarm Optimization - APSO

Zhan *et al.* proposed the Adaptive Particle Swarm Optimization (APSO) in 2009 [ZZLC09]. This variation of PSO was proposed to solve two inefficiencies of the standard PSO algorithm: low convergence velocity and incapability to avoid local minima. The Adaptive PSO aims to achieve these goals with a systematic adaptation of the parameters and the use of an elitist learning strategy.

Basically, the APSO consists in a loop with the following steps: (i) evaluate and estimate the distribution of particles in search space through the metric proposed by authors, called evolutionary factor; (ii) classify the evolutionary state of the swarm; (iii) determinate the acceleration coefficients based on the evolutionary state (improving the convergence velocity); (iv) maintain the diversity with elitist learning strategy; and, (v) adapt the inertial factor, increasing the efficiency of the search process.

2.2.1 Estimation of Evolutionary Factor

One needs to estimate the evolutionary factor in order to control the adaptation process of the APSO. To accomplish this, it is necessary to evaluate the distribution

Algorithm 1: Pseudocode of the PSO algorithm.

```
1  Initialize particles of swarm;
2  Initialize inertial factor;
3  Initialize social and cognitive memory;
4  while the stop criterion is not achieved do
5      for each particle do
6          Update the velocity according to Equation (2.1);
7          Update the position according to Equation (2.2);
8          Evaluate the position using the objective function of problem;
9          if current position is better than cognitive memory then
10             Update the cognitive memory;
11         end
12         if current position is better than social memory then
13             Update the social memory;
14         end
15     end
16     Update inertial factor according to Equation (2.3);
17 end
18 return the social memory.
```

of the particles in the search space along the iterations. One needs to calculate the average Euclidian distance of each particle to the other particles of the swarm. The average distance (d_i) between particle i and the rest of the swarm is evaluated by Equation (2.4).

$$d_i = \frac{1}{N-1} \sum_{j=1, j\neq i}^{N} \sqrt{\sum_{k=1}^{D} (x_i^k - x_j^k)}, \tag{2.4}$$

in which N is the number of particles in the swarm and D is the number of dimensions.

Then, the evolutionary factor (f_{evol}) is calculated according to Equation (2.5).

$$f_{evol} = \frac{d_g - d_{min}}{d_{max} - d_{min}} \quad \in \quad [0, 1], \tag{2.5}$$

in which d_g is the average distance between the best particle of the swarm and the rest of the swarm, d_{min} and d_{max} are the smallest and the biggest average distance among all particles, respectively. To avoid $d_{min} = d_{max}$, it is necessary to have at least three particles in the population.

f_{evol} is a number in the interval $[0, 1]$. If the swarm is close to the best particle, then the evolutionary factor is close to 0. On the other hand, if the particles of swarm are completely spread over the search space, then the evolutionary factor is close to 1.

2.2.2 Classification of Evolutionary State

The original proposal of the APSO algorithm classifies the swarm in an evolutionary state using fuzzy rules. The evolutionary state is chosen at each iteration based on the membership function with higher value. Figure 2.2 presents the four membership functions for the four evolutionary states: Convergence, Exploitation, Exploration and Jumping out.

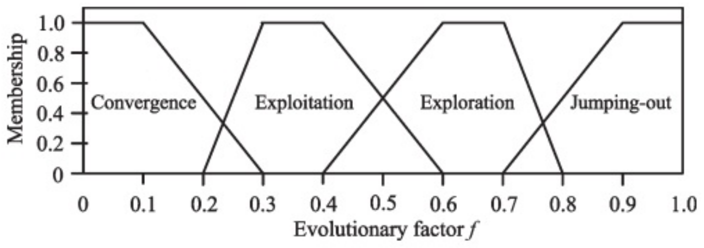


Figure 2.2: Fuzzy membership functions for the APSO algorithm.

Convergence State: it occurs when the value of the evolutionary factor is minimal and the particles are very close of the best particle. This means that the algorithm found a good region of the search space and then it is currently refining the solutions. We calculate the fuzzy value of this state according to Equation (2.6).

$$S_{convergence}(f_{evol}) = \begin{cases} 1, & \text{if } 0.0 \leq f_{evol} \leq 0.1, \\ -5f_{evol} + 1.5, & \text{if } 0.1 < f_{evol} \leq 0.3, \\ 0, & \text{if } 0.3 < f_{evol} \leq 1.0. \end{cases} \tag{2.6}$$

Exploitation State: it occurs when the value of the evolutionary factor is shrunk and the particles are near to the best particle, it indicates that the swarm found a good region of the search space. The membership function of this state is defined according to Equation (2.7).

$$S_{exploitation}(f_{evol}) = \begin{cases} 0, & \text{if } 0.0 \leq f_{evol} \leq 0.2, \\ 10f_{evol} - 2, & \text{if } 0.2 < f_{evol} \leq 0.3, \\ 1, & \text{if } 0.3 < f_{evol} \leq 0.4, \\ -5f_{evol} + 3, & \text{if } 0.4 < f_{evol} \leq 0.6, \\ 0, & \text{if } 0.6 < f_{evol} \leq 1.0. \end{cases} \tag{2.7}$$

Exploration State: it occurs when the value of the evolutionary factor is medium to large and the particles have a medium or large distant of the best particle. In this case, the algorithm is trying to find a good region of the search space. We calculate the fuzzy value of this state according to Equation (2.8).

$$S_{exploration}(f_{evol}) = \begin{cases} 0, & \text{if } 0.0 \leq f_{evol} \leq 0.4, \\ 5f_{evol} - 2, & \text{if } 0.4 < f_{evol} \leq 0.6, \\ 1, & \text{if } 0.6 < f_{evol} \leq 0.7, \\ -10f_{evol} + 8, & \text{if } 0.7 < f_{evol} \leq 0.8, \\ 0, & \text{if } 0.8 < f_{evol} \leq 1.0. \end{cases} \tag{2.8}$$

Jumping out State: it occurs when the evolutionary factor presents large values. This means that the APSO is jumping out of a local optimum to a new region, in other words, the globally best particle is distinctively away from the cluster of the swarm. The membership function of this state is defined according to Equation (2.9).

$$S_{jumping_out}(f_{evol}) = \begin{cases} 0, & \text{if } 0.0 \leq f_{evol} \leq 0.7, \\ 5f_{evol} - 3.5, & \text{if } 0.7 < f_{evol} \leq 0.9, \\ 1, & \text{if } 0.9 < f_{evol} \leq 1.0. \end{cases} \tag{2.9}$$

2.2.3 Adaptation of Acceleration Coefficients

The next step is to update the acceleration coefficients c_1 and c_2, which are initialized with values equal to 2.0 and are updated according to the current value of f_{evol}. The rules to update the coefficients are depicted in Table 2.1.

Table 2.1: Strategies for the control of acceleration coefficients c_1 and c_2.

Evolutionary State	c_1	c_2
Convergence	Increase slightly	Increase slightly
Exploitation	Increase slightly	Decrease slightly
Exploration	Increase	Decrease
Jumping out	Decrease	Increase

In the *Convergence* state, the c_1 value is slightly increased and the c_2 value slightly is increased. In this state, the swarm is located at a minima, and, hence, the influence of social coefficient should be prioritized to guide other particles to this region. Thus, the value of c_2 should be increased. On the other hand, the value of the cognitive coefficient should be decreased to guarantee the swarm to converge faster. The consequence of this strategy is a premature saturation of the cognitive and the social coefficients to their lower and upper bounds, respectively. Thus, the swarm will strongly be attracted by the current best region, causing premature convergence, which is unsuitable if the current best region is just a local optimum. To avoid this, both c_1 and c_2 are slightly increased.

In the *Exploitation* state, the c_1 value is slightly increased and the c_2 value is slightly decreased. In the Exploitation state, each particle uses local information and the swarm form groups in potential local optimal regions, that were identified by the historical best position of each particle. Hence, c_1 is slowly increased and maintains a relatively large value to predominate the search and exploitation around the respective cognitive memories. Probably, the social memory is still not present in the global optimal region. Therefore, decreasing c_2 slowly can avoid the lock in local optimal. Furthermore, an exploitation state in more likely to occur after an exploration state and before a convergence state. Hence, changing directions for c_1 and c_2 should be slightly altered from the exploration state to the convergence state [ZZLC09].

In the *Exploration* state, the c_1 value is increased and the c_2 value is decreased. In this state, is essential the swarm should explore so many optima search regions as possible. hence, increasing c_1 and decreasing c_2 can help each particle explores individually and achieve its own historical best positions. In this case, we aim to avoid that the current best particle of swarm get stucked in a local optimum.

In the *Jumping out* state, the c_1 value is decreased and the c_2 value is increased. When the best particle is jumping out of the local optimum toward a better optimum, it is very likely to be far away from the core of the swarm in presented in the Convergence state. As soon as this new region is found by a particle, which becomes the (possibly new) guide, others should follow it and achieve this new region as fast as possible. To guarantee this behavior of the swarm, a large c_2 value together with a relatively small c_1 value is more indicated.

The incremented or decremented value of acceleration coefficients is called acceleration rate (δ). In the APSO algorithm, this value is a random number uniformly generated among the interval [0.05, 0.1]. The term "slightly" found in strategies of Convergence and Exploitation states, implies in the use of acceleration rate equal to 50% ($\delta \cdot 0.5$).

In the coefficients adaptation process, they can achieve huge or small values, generating unstable moments in search process. To solve this, the authors of APSO proposed a normalization of acceleration coefficient values. The lower and upper boundaries of acceleration coefficients are $c_{min} = 1.5$ and $c_{max} = 2.5$, respectively. If the sum of c_1 and c_2 is larger than 4.0, the acceleration coefficients should be normalized according to Equation (2.10).

$$c_i = \frac{c_i(c_{min} + c_{max})}{c_1 + c_2}, \quad i = 1, 2. \tag{2.10}$$

2.2.4 Elitist Learning Strategy

The third step is the application of a operator in order to generate diversity. The authors named it as Learning Strategy using Elitism and this mechanism aims to improve the global search capability of the algorithm. It was first proposed to be applied just in the best particle during the Convergence state, generating the Jumping out state.

The reason for this operator is because the best particle does not have exemplars to follow. As a means to improve its own, a perturbation was developed to help the best particle to push itself out of a local minima to a potential better search region. If the new region is better, then the rest of swarm will follow quickly the leader in order to converge to this new search region.

This is a type of greedy local search applied only in one dimension (d) of the current best particle ($\vec{n}_i(t)$) of the swarm aiming to allow this particle to escape from a local optimum. All the dimensions have the same probability to be chosen. The Gaussian mutation is generated according to Equation (2.11).

$$\vec{n}_i(t+1) = \vec{n}_i(t) + (X^d_{max} - X^d_{min})G(\mu, \sigma^2), \tag{2.11}$$

in which (X^d_{max}, X^d_{min}) are the boundaries of search space, $G(\mu, \sigma^2)$ is a random number generated by Gaussian distribution with a zero mean μ and standard deviation σ, which is called the elitist learning rate. The authors suggested that σ be linearly decreased with the iterations number and is calculated by Equation (2.12).

$$\sigma = \sigma_{max} - \left[(\sigma_{max} - \sigma_{min})\frac{g(t)}{g_{final}}\right], \tag{2.12}$$

in which $(\sigma_{max}, \sigma_{min})$ are the boundaries of σ, $g(t)$ is the current iterations and g_{final} is the total number of iterations. In empirical tests, Zhan *et al.* proposed to use $\sigma = 1.0$ in the beginning of the simulations, with the objective to escape from a local optimum and decrease it to $\sigma = 0.1$ at the end of the simulation, to refine the found solutions.

The position of the best particle is updated only if the new position found by the operator is better than the previous one. Otherwise, the new position will replace the worst particle in the swarm.

2.2.5 Adaptation of Inertial Factor

In the last step, the inertial factor ω is updated using equation (2.13) in order to auto-adapt the exploration-exploitation ability of the swarm.

$$\omega(f_{evol}) = \frac{1}{1 + 1.5e^{-2.6 f_{evol}}} \quad \in \quad [0.4; 0.9], \quad \forall f_{evol} \quad \in \quad [0,1]. \tag{2.13}$$

2.2.6 Pseudocode of the APSO Algorithm

The pseudocode of the APSO algorithm is shown in Algorithm 2. The APSO algorithm has the same main steps that are found in the PSO algorithm. However, one can observe the adaptation of parameters (acceleration coefficients and inertial factor) in the lines 3 to 20. In lines 3 to 6, one can observe the steps needed to estimate the evolutionary factor. The classification of evolutionary state of swarm can be observed in lines 7 and 8. In the lines 9 and 10, one can observe the adaptation of acceleration coefficients. In line 11, the adaptation of inertial factor is executed. In the lines 12 to 19, one can observe the execution of elitist learning strategy.

2.3 Clan Particle Swarm Optimization - ClanPSO

Clans are groups of individuals united by a kinship based on a lineage, for example. Some clans stipulate a common ancestor to become the leader of the clan,

Algorithm 2: Pseudocode of the APSO algorithm.

```
1  Initialize particles of swarm;
2  while the stop criterion is not achieved do
3      for each particle do
4          Calculate the average distance according to Equation (2.4);
5      end
6      Calculate the evolutionary factor according to Equation (2.5);
7      Calculate the membership function according to Equations (2.6), (2.7),
       (2.8) and (2.9);
8      Classify the swarm in the evolutionary state;
9      Adapt the acceleration coefficients according to Table 2.1;
10     Normalize the acceleration coefficients according to Equation (2.10);
11     Update the inertial factor according to Equation (2.13);
12     if classified on Convergence state then
13         Generate a new position using the Equations (2.11) and (2.12);
14         if new position is better than the best cognitive memory then
15             Update the position and cognitive memory of the best particle;
16         end
17         else
18             Update the position and cognitive memory of the worst particle;
19         end
20     end
21     for each particle do
22         Update the social memory in neighborhood;
23         Update the velocity according to Equation (2.1);
24         Update the position according to Equation (2.2);
25         Evaluate the position using the objective function of problem;
26         if current position is better than cognitive memory then
27             Update the cognitive memory;
28         end
29         if current position is better than social memory then
30             Update the social memory;
31         end
32     end
33 end
34 return the social memory.
```

and every individual of the clan will be guided by this leader. Incorporating this characteristic, a topology was proposed for improving the PSO algorithm performance, the Clan Particle Swarm Optimization [CBF08]. This topology consists in a set of clan, where each clan is sub-swarm and uses a fully-connected structure to share information. The structure presented in Figure 2.3 is an example with four clans (A, B, C and D) of five particles.

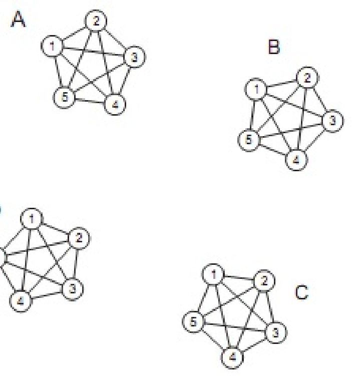


Figure 2.3: Example of division of the population in sub-swarms.

Other approach was proposed that enables the migration of particles among clans and some improvements were reached for some benchmark function [BFCFdM09]. However, there are some problems in this case, such as possibility of empty clans.

In this algorithm, the population size is divided according to the number of particle per clan (N_{pc}). For each iteration, each one of clans performs a search using a classical PSO and selects the particle that had achieved the best position of the entire clan. This particle is called the leader and this process of marking is called delegation. After the definition of all leaders, we form a new swarm with only the leaders of clans and we execute the PSO algorithm just with the leaders. This process is called conference of leaders. We can observe in more details these execution steps of the ClanPSO algorithm.

2.3.1 Delegation of Leaders

Is similarly to complex social behavior, with several clans with various leaders. The definition of a leader in clan is a marking process the best particle in the swarm, for example in Figure 2.4. The delegation process is based in the global information exchange mechanism inside each clan, which uses $\vec{n_{best}}$ information to delegate the leader.

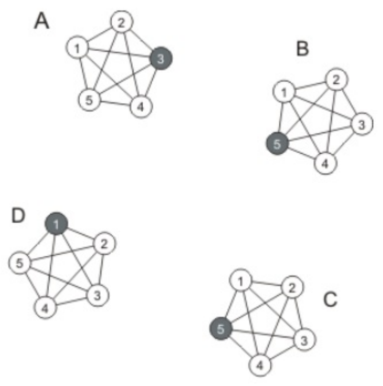


Figure 2.4: Example of definition of clans leaders (A, B, C and D).

2.3.2 Conference of Leaders

After the delegation, the leaders need to adjust their positions based on the best leader. This second step consists in execution of PSO only with the leaders, and is called conference of leaders. The conference can be performed using either the star (observe Figure 2.5(a)) or ring (observe Figure 2.5(b)) topology.

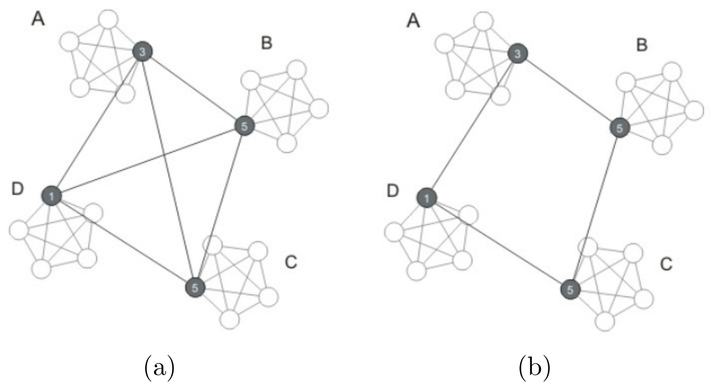


Figure 2.5: Types of conference in the ClanPSO: (a) global conference and (b) local conference.

The topology to be used depends on the kind of problem to be solved. When the star topology is used, the information is spread faster between leaders through of global information share, guaranteeing a better exploitation ability. When is used ring topology, the communication between leaders is slower, thus the algorithm has great exploration ability. The convergence velocity is smaller than in star topology, but the quality of the solutions are often better.

2.3.3 The Information shared within the clans

After the leaders conference, the leaders will return to its respective clans and the new information acquired in the process will be widely used inside each clan to adjust the other particles. Indirectly, allows that all other particles be guided by the best position found within the entire topology. Through indirect communication, a foreign leader does not directly influence another clan, and then preserves the exploration capacity of clans.

2.3.4 Pseudocode of the ClanPSO Algorithm

The pseudocode of the ClanPSO algorithm is shown in Algorithm 3. In the lines 4 to 14, one can observe the PSO execution within of each clan. In lines 17 to 28, one can observe the leaders conference. We can observe the leaders delegation in the lines 5 and 16, and still the line 5, the return of information from the leaders conference.

Algorithm 3: Pseudocode of the ClanPSO algorithm.

```
Initialize the particles with random positions and velocities;
Group the particles in clans with global topology;
while the stop criterion is not achieved do
    for each clan do
        Find the best particle and mark as leader;
        Update the velocity according to Equation (2.1);
        Update the position according to Equation (2.2);
        Evaluate the position using objective function of the problem;
        if new position is better than cognitive memory then
            Update cognitive memory;
        end
        if new position is better than social memory then
            Update social memory;
        end
    end
    Add clan leader in the leaders list;
    for each leader particle of the list do
        Find the best particle;
        Update the velocity according to Equation (2.1);
        Update the position according to Equation (2.2);
        Evaluate the position using objective function of the problem;
        if new position is better then
            Update cognitive memory;
        end
        if new position is better than social memory then
            Update social memory;
        end
    end
end
return the social memory.
```

2.4 Clan Adaptive Particle Swarm Optimization - ClanAPSO

The Clan Adaptive Particle Swarm Optimization (ClanAPSO) was developed in 2011 [PLNBF11]. This algorithm was proposed by aggregation of the process of systematic adaptation of parameters with a search strategy of multi-swarm. This technique was inspired in the clans topology presents in the ClanPSO algorithm combined to the concepts of parameters adaptation and elitist strategy found in the APSO algorithm.

The ClanAPSO algorithm presents an adaptation process in each swarm in a multi swarm system, this allows that sub-swarms can perform different types of search by iteration of the algorithm, *i.e.* it is possible to perform exploitation and exploration search simultaneously in distinct regions of the search space. The concept of clans is interesting because it allows to identify local leaders of groups of particles (clans) and carry out the information share between clans indirectly. The indirect communication between clans promotes improvement the search process because it is a distributed process of spreading information.

There are two versions of the ClanAPSO algorithm. The original version [PLNBF11] executes the APSO algorithm within the clans and also execute the APSO in the conference of leaders. The proposed second version [VRBF11] also includes the adaptation capability within the clans, but in the leaders conference is running the PSO algorithm.

2.4.1 Pseudocode of the ClanAPSO Algorithm

The simplified pseudocode of the ClanAPSO algorithm is shown in Algorithm 4. In the lines 3 to 6, the APSO execution within of each clan is presented. In the lines 7 and 8, one can observe the conference of leaders, where they can be executed by the APSO algorithm or the PSO algorithm. We can observe the leaders delegation in the line 5.

Algorithm 4: Pseudocode of the ClanAPSO algorithm.

```
1  Initialize the particles with random positions and velocities;
2  while the stop criterion is not achieved do
3      for each clan do
4          Execute APSO within the clans;
5          Delegate leader;
6      end
7      Create conference of leaders;
8      Execute APSO (or PSO) with the leaders;
9  end
10 return the best particle.
```

2.5 Artificial Bee Colony - ABC

The ABC algorithm was proposed by Karaboga in 2005 [Kar05]. ABC is an algorithm of search and optimization modeled by the behavior of honey bees [BK06] [KA07] [KA09a] [KA09b] [ZAZ11]. The bee colony is one of the natural societies with the most specialized social divisions. In the simplified model assumed by the ABC, the bee colony is composed by three types of bees: employed bees, onlookers and scouts. Initially, the swarm is divided in equal parts of employed bees and onlookers bees. The employed bees are those that go to the food source explored by herself, there is a food source to each employed bee. The bees that wait in the hive and decide to exploit a food source depending on the information shared by the employed bees are called onlooker bees. The onlookers are guided bees. When the food source does not improve its quality, the associated employed bee is now scout bee, which is responsible to find a new valuable food source. We will call them guide bees in exploration mode.

Basically, the steps of the algorithm executed in each iteration are: (*i*) the employed bees explore its respective food sources; (*ii*) it is determined the quality of food sources, which are shared to onlooker bees; (*iii*) the onlooker bees choose a food source and help to explore the food source selected; (*iv*) if the food source does not improve, the associated employed bee becomes scout bee. The scout bees find randomly new source food and determine its quality. In the end of each iteration, the best food source is saved until the end of the algorithm execution.

A food source represents a potential solution for the problem to be solved. The nectar quantity of a food source determines the solution quality of that food source. The onlooker bees choose the best food source using the selection method by roulette wheel.

The scout bees are explorers of the hive. They do not follow orientations to search food source, then this type of bees aims to find new food sources. The quality of these food source is not relevant (the most of cases is medium or low). However, in some cases, the scout bees find good food sources, that were unknown. The low search cost makes this mechanism of maintain diversity an attractive option.

We can observe that the ABC algorithm has exploitation-exploration tradeoff because the employed and onlookers bees has exploitation capability and scout bees has exploration capability.

2.5.1 Quality of Food Source

The quality of food source is calculated based on objective function value of problem according to Equation (2.14) [MM12].

$$Q[\vec{x}(t)] = \begin{cases} \frac{1}{f[\vec{x}(t)]+1}, & \text{if } f[\vec{x}(t)] \geq 0, \\ 1 + abs f[\vec{x}(t)] + 1, & \text{if } f[\vec{x}(t)] < 0, \end{cases} \tag{2.14}$$

in which $\vec{x}(t)$ is food source position, abs is value of absolute function, $Q[\vec{x}(t)]$ is the quantity of food source nectar presents in position $\vec{x}(t)$ and $f[\vec{x}(t)]$ is the value of the objective function in position $\vec{x}(t)$.

2.5.2 Update of Food Source

In the algorithm, the search space of the problem is D-dimensional. The number of the employed bees and the onlookers are the same and for every food source, there is only one employed bee per food source. In other words, the size of employed and onlooker bees is equal SN (the number of food sources). Each ith source food associated to the employed bee will be optimized according to Equation (2.15).

$$v_{id} = x_{id} + r_{id}(x_{id} - x_{kd}), \tag{2.15}$$

in which $d = 1, 2, ..., D$ is the number of dimensions, r_{id} is a random generalized real number within the range [-1,1], $k = 1, 2, ..., SN$ is a randomly selected index number in the colony, it has to be different from the i. The new solution (v_{id}) is compared with the previous one (x_{id}), and the better one should be stored.

2.5.3 Choose of Food Source by Onlookers Bees

Next, the onlooker bee needs to select one of the food sources explored by the employed bees. The probability for each food source to be selected by the onlooker bee is:

$$p_i = \frac{fit_i}{\sum_j^{SN} fit_j}, \tag{2.16}$$

in which p_i is the probability to select the food source i, which is proportional to the quality of the food source, fit_i is the fitness of the position x_{id}. Each onlooker bee searchers for a new solution in the selected food source by using Equation (2.15).

2.5.4 Stagnation of Food Source

At each loop, the food source are evaluated and if the fitness of a source does not improve after a predetermined number of steps (called *MaxTrial*), then it is abandoned. The employed bee associated with it becomes a scout and replaces the food source with a new one, through the random search given by Equation (2.17).

$$x_{id} = x_d^{min} + r(x_d^{max} - x_d^{min}). \tag{2.17}$$

in which r is random number selected in range [0,1], and x_d^{max} and x_d^{min} are lower and upper borders in the d^{th} dimension of the search space.

2.5.5 Pseudocode of the ABC Algorithm

The pseudocode of the ABC algorithm is shown in Algorithm 5. In the pseudocode, we can observe the employed bees movement in lines 3 to 13. The probability used to selection of employed bees by onlooker bees can be observed in the line 16. Then, the onlooker bees movement can be observed in lines 18 to 28. If some food source does not improve, so the employed bee becomes a scout bee. This movement can be visualized in lines 29 to 35.

2.6 Fish School Search - FSS

The Fish School Search (FSS) is a computational intelligence technique inspired in social behavior of schools of fish developed by Bastos-Filho and Lima-Neto in

Algorithm 5: Pseudocode of the ABC algorithm.

```
Initialize the food sources in random positions;
while the stop criterion is not achieved do
    for each employed bee do
        Determine a different position randomly;
        Update the position according to Equation (2.15);
        Evaluate the position using the objective function of problem;
        if the new position is better then
            Update the position of associated food source;
        end
        else
            Increase the stagnation counter of food source;
        end
        Calculate the quality of new food source position according to
        Equation (2.14);
    end
    for each employed bee do
        Calculate the probability od roulette wheel selection using the
        Equation (2.16);
    end
    for each onlooker bee do
        Determine two different positions chosen by roulette;
        Update the position according to Equation (2.15);
        Evaluate the position using the objective function of problem;
        if the new position is better then
            Update the position of associated food source;
        end
        else
            Increase the stagnation counter of food source;
        end
    end
    for each employed bee do
        if stagnation counter achieved the threshold then
            Generate a new position of associated food source randomly;
            Restart the stagnation counter of associated food source;
        end
    end
end
return the best food source.
```

2007 [BFLNL+08] [BFNS+09]. It was conceived to solve search problems and was based in the gregarious behavior present in some fish species, with the objective to improve the survivability of the entire group through the mutual protection and synergy to perform collective tasks [LBFN+12].

In the FSS algorithm, the search space, called aquarium, is limited region of objective function, the population is called school of fish and each fish has a weight. The weight of the fish represents the success of search process. Each position in the search space represents a possible solution of the problem.

During the execution, the operators of the algorithm are executed sequentially by updating the positions and weights of each fish. The FSS algorithm has four operators: individual movement (responsible by local search), feeding (indicator of success of search process), collective-instinctive movement (generates the displacement of school of fish) and collective-volitive movement (controls the exploitation-exploration granularity).

2.6.1 Individual Movement Operator

Initially, the fish try to find food and for this, the fish realize individual movements in the search space. The individual movement is executed by each fish in the school (S) in the beginning of each iteration. Each fish chooses a new position in its neighborhood and then, this new position is evaluated using the objective function of problem. The individual movement operator is determined according to Equation (2.18) for each dimension.

$$v_{id}(t) = x_{id}(t) + rand[-1,1] \cdot step_{ind}, \tag{2.18}$$

in which $\vec{v}_i(t)$ is the candidate position of fish i, $\vec{x}_i(t)$ is the current position of the fish i, $rand[-1,1]$ is a random number generated by an uniform distribution in the range [-1,1] and d is a number of dimensions. The $step_{ind}$ is a percentage of search space amplitude in dimension determined.

The $step_{ind}$ decreases linearly along the iterations according to Equation (2.19), so the exploitation capability increases during search process.

$$step_{ind} = step_{ind\_initial} - \left[(step_{ind\_initial} - step_{ind\_end})\frac{g(t)}{g_{end}}\right], \tag{2.19}$$

in which $step_{ind\_initial}$ is the individual step in the beginning of the algorithm execution, $step_{ind\_end}$ is the individual step in the final of the algorithm execution, $g(t)$ is the current iteration and g_{end} is the total number of iterations.

After the calculation of the candidate position, the movement just occurs if the new position has better fitness than the old one.

2.6.2 Feeding Operator

The fish weight can grow or decrease, depending on its success or failure in the search for food, when the individual movement is realized. In each iteration, the fish weight is updated according to Equation (2.20).

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{max(\Delta f)}, \tag{2.20}$$

in which $W_i(t)$ is the weight of the fish, Δf_i is the difference between the fitness value of new position ($f[\vec{x}(t+1)]$) and the fitness value of the current position for each fish ($f[\vec{x}(t)]$), this value is calculated according to Equation (2.21). The $max(\Delta f)$ is the maximum value of these differences in the current iteration.

$$\Delta f = f[\vec{x}(t+1)] - f[\vec{x}(t)]. \tag{2.21}$$

In order to avoid a explosion state of the weight values, the algorithm has a maximum value, called weight scale (W_{scale}), that is defined to limit the weight of fish. The initial weight for each fish is equal to $\frac{W_{scale}}{2}$.

2.6.3 Collective-Instinctive Movement Operator

After the individual movement, the fish are evaluated and is observed if were successful in the food search or not. Therefore, the school should move based in the successful fish, *i.e.* is a global movement in direction, probably, to richer region of food.

The position of all the fish are updated with this movement according to Equation (2.22).

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \frac{\sum_{i=1}^{N} \Delta\vec{x}_{ind_i}\Delta f_i}{\sum_{i=1}^{N} \Delta f_i}, \tag{2.22}$$

in which $\Delta\vec{x}_{ind_i}$ is the displacement of the fish i due to the individual movement in the FSS cycle. One must observe that $\Delta\vec{x}_{ind_i} = 0$ for fish that did not execute the individual movement [LBFN$^+$12].

2.6.4 Collective-Volitive Movement Operator

After the other two movements, the school of fish executes the collective-volitive movement. This movement is global able to make the school fish expand or contract. If the fish school search has been successful, *i.e.* its weight is increasing, the radius of the school should contract to better exploitation capability; if not, it should expand, allowing better exploration capability. Thus, this operator increases the capacity to auto-regulate the exploration-exploitation granularity [LBFN$^+$12].

To realize the school dilation or contraction in each fish position is necessary the school centroid, which can be evaluated by using the Equation (2.23).

$$\vec{B}(t) = \frac{\sum_{i=1}^{N} \vec{x}_i W_i(t)}{\sum_{i=1}^{N} W_i(t)}. \tag{2.23}$$

The movement is executed according to Equation (2.24). To the fish school expansion, we use signal '+' and to the fish school contraction, we use signal '−'.

$$\vec{x}_i(t+1) = \vec{x}_i(t) \pm step_{vol} r_1 \frac{\vec{x}_i(t) - \vec{B}(t)}{d(\vec{x}_i(t), \vec{B}(t))}, \tag{2.24}$$

in which $step_{vol}$ is called volitive step, r_1 is a random number generated by uniform probability density function in the range [0,1]. $d(\vec{x}_i(t), \vec{B}(t))$ calculates the euclidian distance between the particle i and the centroid.

The $step_{vol}$ value decreases linearly along the iterations of the algorithm according to Equation (2.25). Thus, the algorithm initializes with an exploration ability and changes to an exploitation mode. The parameter value is defined as a percentage of the search space range and is bounded by two parameters ($step_{vol\_initial}$ and $step_{vol\_end}$).

$$step_{vol}(t) = step_{vol\_initial} - \left[(step_{vol\_initial} - step_{vol\_end})\frac{g(t)}{g_{end}}\right], \tag{2.25}$$

in which $g(t)$ is the current iteration and g_{end} is the total number of iterations. Usually, $step_{vol} = step_{ind}$.

After the update of all the fish, is necessary to evaluate the school fish with objective function of problem, thus they will be prepared for the next iteration.

2.6.5 Pseudocode of the FSS Algorithm

The pseudocode of the FSS algorithm is shown in Algorithm 6. In the lines 3 to 9, we can observe the individual movement that each fish realizes. In the line 17, one can observe the collective-instinctive movement and in the lines 19 to 26, the collective-volitive movement. In the lines 5 and 27, one can observe the evaluation of objective function of problem.

2.7 Discussion about the Swarm Intelligence Algorithms

This chapter described the main swarm intelligence algorithms that were essential to development of proposed algorithm in this dissertation.

Particle Swarm Optimization - PSO: this is standard version of the algorithm. It is the most known swarm intelligence algorithm, very simple and easy deployment. This algorithm is very used in unimodal problems due to its convergence ability, but is not recommended in multimodal problems because the algorithm has limitations to maintain diversity of swarm. Other approaches were developed to solve these limitations [Cle99] [CK02] as new communications topologies [CBF08] [BFCFdM09] [PLNBF11] [BFCMC08].

Adaptive Particle Swarm Optimization - APSO: was proposed to solve two limitations of original PSO, the low convergence velocity and incapability to escape of local minimum. This approach presents a systematic scheme to update the PSO parameters based on an evolutionary factor, that is calculated through distribution of particles in search space. This scheme allows a better control of convergence velocity and ability to escape of local minimum.

Clan Particle Swarm Optimization - ClanPSO: a new topology for PSO algorithm compound by particle clans which communicate in conference to share

Algorithm 6: Pseudocode of the FSS algorithm.

```
1  Initialize the population in random positions;
2  while the stop criterion is not achieved do
3      for each fish do
4          Execute individual movement according to Equation (2.18);
5          Evaluate position using the objective function of problem;
6          if new position is better then
7              Update the fish memory;
8          end
9      end
10     Update the individual step according to Equation (2.19);
11     Calculate the population weight before to adjust it;
12     for each fish do
13         Adjust the weight according to Equation (2.20);
14     end
15     Calculate the population weight after of the adjust of the all the fish;
16     for each fish do
17         Execute the instinctive movement according to Equation (2.22);
18     end
19     Calculate the centroid of the swarm according to Equation (2.23);
20     for each fish do
21         if swarm increased its weight then
22             Execute volitive movement of contraction according to Equation
               (2.24);
23         end
24         else
25             Execute volitive movement of dispersion according to Equation
               (2.24);
26         end
27         Evaluate the position using objective function of problem;
28         if new position is better then
29             Update the fish memory;
30         end
31     end
32     Update the volitive step according to Equation (2.25);
33 end
34 return the best memory of school of fish.
```

information. The exchange of information indirectly between the clans can offer improved performance of the algorithm, avoiding premature convergence. However, setting the number of particles per clans and clan may depend on the problem.

Clan Adaptive Particle Swarm Optimization - ClanAPSO: was introduced the APSO algorithm concepts in each clan. It was used another independent APSO (or PSO) to perform the conference of leaders. Thus, each clan adapt itself independently and, as a consequence, the algorithm will avoid to exploit excessively the same spot or converge prematurely.

Artificial Bee Colony - ABC: was inspired in honey bees behavior and has the ability to explore richer food sources with the onlooker (guided bees) and employed (guide bees in exploitation mode) bees. This exploitation ability is controlled with stagnation counter and the algorithm maintains the diversity of bee colony with the scout bees (guide bees in exploration mode).

Fish School Search - FSS: was inspired in school of fish behavior and has two global movements. The first movement moves the school in direction the fish that achieved the best results. The second movement regulates the search granularity, allowing the contraction or expansion of school. Due the presence of two fitness evaluation, this algorithm is slower than other approaches, such as PSO.

Table 4.2 shows the number of parameters of each algorithm that need to setup. Some parameters present standard values, but the researcher can change it.

Table 2.2: The number of parameters of each algorithm.

Algorithm	**Number of Parameters**	**Parameters**
PSO	5	N, c_1, c_2, ω_{min}, ω_{max}
APSO	5	N, c_1, c_2, ω_{min}, ω_{max}
ClanPSO	6	N, c_1, c_2, ω_{min}, ω_{max}, N_{pc}
ClanAPSO	6	N, c_1, c_2, ω_{min}, ω_{max}, N_{pc}
ABC	2	$2 \cdot SN$, $MaxTrial$
FSS	6	S, W_{scale}, $step_{vol\_initial}$, $step_{vol\_final}$, $step_{ind\_initial}$, $step_{ind\_final}$

This dissertation proposes to combine two well known swarm intelligence algorithms: the APSO algorithm and the ABC algorithm. The proposed algorithm has a operator based on bees behavior (the ABC algorithm) to generate diversity for Particle Swarm Optimization with adaptive behavior.

Chapter 3

Our Proposal: An Operator based on Artificial Bees to Generate Diversity for Particle Swarm Optimization

This chapter describes the operator proposed that was included in the APSO algorithm. This combination generated a algorithm proposed to be applied in problem very complex with search space of high dimensionality. The proposed algorithm, called *Adaptive Bee and Particle Swarm Optimization* (ABeePSO), combines two classic well known swarm intelligence algorithms: APSO and ABC. The use of APSO is due to better convergence capability and some deficiencies were mitigated of the original PSO, with adaptive mechanisms based on the evolutionary factor. To maintain the diversity of swarm, we use the ABC algorithm because it performs the behavior change of employed bee to scout bee within the bee colony and allows the ability to escape from local minimum. Actually, the ABC algorithm was modified for this proposal. It used the calculation of centroid presents in the FSS algorithm to spread the bees in the search space.

3.1 Motivation

The APSO algorithm presents a more sophisticated mechanism to adapt the acceleration coefficients of the velocity equation based on the evolutionary factor, that was demonstrated an improvement in the convergence capability. However, the operator used to generate diversity (during the *Jumping out* state) can be improved. The APSO Learning Strategy moves one particle per iteration and it does not provide enough diversity to solve more complicated problems (problems of high dimensionality).

However, some other swarm-based algorithms can deal more properly with this problem, such as the ABC that presents the capability to generate diversity when the guide bees are in the exploration mode. In this paper, we propose an approach to generate diversity by using the ABC when the APSO stagnates, i.e., to include the exploration ability of the ABC algorithm instead of using the Learning Strategy used in APSO. The swarm switches its behavior depending on a measurement of the diversity of the entire swarm.

3.2 Adaptive Bee and Particle Swarm Optimization

In our proposal, when the evolutionary state of the swarm is *Convergence* the ABC algorithm is executed. Actually, it is an operator based on the ABC algorithm, that has the following steps: (i) guide bees movement; (ii) guided bees movement; and, (iii) classification of evolutionary state.

3.2.1 Guide Bees Movement

During the ABC execution, the first movement is performed by the guide bees, that it was inspired in the collective-volitive movement of FSS algorithm because used the centroid concept and expansion or contraction of the swarm in the search process (found in the subsection 2.6.4). This movement guarantees the expansion of at least half of the swarm.

The swarm with N particles is divided as follows. The $N/2$ particles with better fitness become guide bees and the other particles become guided bees. The i^{th} guide bee will be optimized according to Equation (3.1):

$$v_{id} = x_{id} + step_d.r_i[x_{id} - B_d/d(x_{id}, B_d)], \tag{3.1}$$

in which B_d is the centroid of the guide bees, defined by Equation (3.2), $step_d$ is the vector of dispersion steps of the swarm, determined by Equation (3.3), r_i is a random number generated by an uniform distribution within the range [0,1] and $d(x_{id}, B_d)$ is the Euclidian distance between centroid (B_d) and current position (x_{id}).

$$B_d = \frac{\sum_{i=1}^{N/2} x_{id} fit_i}{\sum_{i=1}^{N/2} fit_i}, \tag{3.2}$$

$$step_d = \{1 - [1/(1 + e^{\alpha(-f_{evol}+\beta)})]\}(x_d^{max} - x_d^{min}), \tag{3.3}$$

where x_d^{max} and x_d^{min} are the lower and upper borders in the d^{th} dimension of the search space, the displacement β indicates the transition point of the logistic function and α is the sigmoid parameter.

Figure 3.1 shows the graph of logistic function of dispersion step, with the values indicated. The reason of the use of logistic function is due to its characteristics. In more details, when the evolutionary factor value is small (that indicates the Convergence or Exploitation state of swarm) the step value $step_d$ is great, thus the swarm will expand faster. However, when the evolutionary factor is higher than 0.5, the step value is small. In this case, it is not necessary a high step value because the swarm is already spread. Thus, this avoids the explosion state of step value.

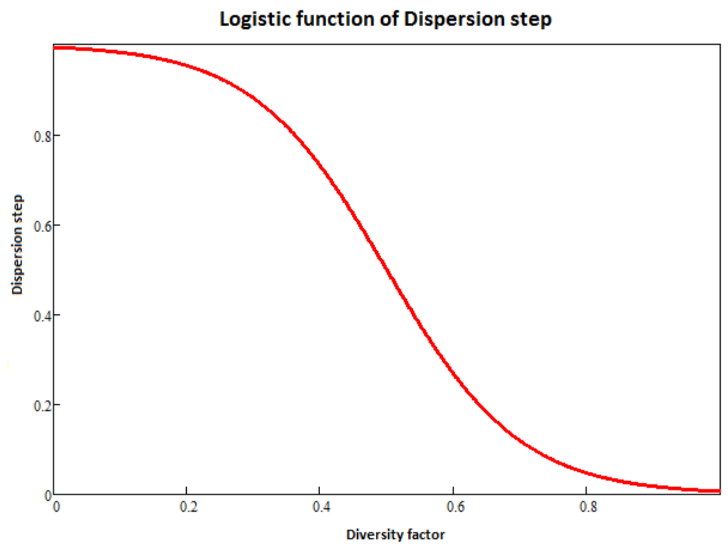


Figure 3.1: Graphic illustration of logistic function of Dispersion step $step_d$.

When the fitness value changes, we update the value of the dispersion according to Equation (3.4). If the new position is better than the current position, the step value is decreased because it found a good region; otherwise, the step value is increased.

$$step_d = step_d \pm f_{evol} step_d. \tag{3.4}$$

3.2.2 Guided Bees Movement

After the guide bee movement, the guided bees need to select one of the food sources, where the probability to choose a food source is given by Equation (2.16). The guided bees movement is realized based on centroid (B_d) of swarm and the food source chosen to follow. Each guided bee searches for a new solution around the food source by using Equation (3.5):

$$v_{id} = x_{id} + r_i(fs_{id} - B_d), \tag{3.5}$$

where r_i is a random number within the range [0,1] and fs_{id} is the food source selected by the guided bee.

3.2.3 Classification of Evolutionary Factor

The next step is to estimate the evolutionary state, such as in the APSO algorithm. However, this estimation just considers the food sources (half of swarm), which are the current potential solutions of the problem.

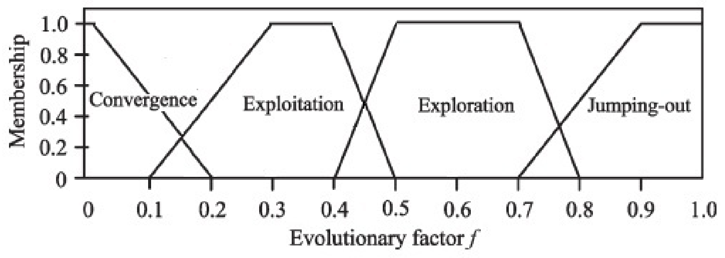


Figure 3.2: Fuzzy membership functions for the ABeePSO algorithm.

Figure 3.2 presents the membership functions used in our proposal. These fuzzy membership functions were defined through parameter analysis found in the Chapter 5.

The classification of evolutionary state was altered in three of four evolutionary states: *Convergence*, *Exploitation* and *Exploration*. Thus, the *Jumping out* state has the same intervals of APSO algorithm. Then, we can observe the new values of intervals that have been changed to adapt the new diversity operator of our proposal.

Convergence State: similar to the APSO algorithm, it occurs when the value of the evolutionary factor is very small and the particles are very close of the best particle. The algorithm is currently refining the solutions. However, the values of this state was altered because this operator spread at least half of the swarm. Thus, the occurrences of this state have to be decreased. We calculate the fuzzy value of this state according to Equation (3.6).

$$S_{convergence}(f_{evol}) = \begin{cases} 1, & \text{if } 0.0 \leq f_{evol} \leq 0.02, \\ (-f_{evol} + 0.2)/0.18, & \text{if } 0.02 < f_{evol} \leq 0.2, \\ 0, & \text{if } 0.2 < f_{evol} \leq 1.0. \end{cases} \tag{3.6}$$

Exploitation State: as a APSO algorithm, it occurs when the value of the evolutionary factor is shrunk and the particles are near to the best particle. The values was modified to avoid the prolonged ABC execution and decrease convergence ability. The membership function of this state is defined according to Equation (3.7).

$$S_{exploitation}(f_{evol}) = \begin{cases} 0, & \text{if } 0.0 \leq f_{evol} \leq 0.1, \\ 5f_{evol} - 0.5, & \text{if } 0.1 < f_{evol} \leq 0.3, \\ 1, & \text{if } 0.3 < f_{evol} \leq 0.4, \\ -10f_{evol} + 5, & \text{if } 0.4 < f_{evol} \leq 0.5, \\ 0, & \text{if } 0.5 < f_{evol} \leq 1.0. \end{cases} \tag{3.7}$$

Exploration State: similar to the APSO algorithm, it occurs when the value of the evolutionary factor is ranging from medium to large and the particles have a medium or large distance of the best particle. The values of this state were altered with the same objective of the other states. We calculate the fuzzy value of this state according to Equation (3.8).

$$S_{exploration}(f_{evol}) = \begin{cases} 0, & \text{if } 0.0 \leq f_{evol} \leq 0.4, \\ 10f_{evol} - 4, & \text{if } 0.4 < f_{evol} \leq 0.5, \\ 1, & \text{if } 0.5 < f_{evol} \leq 0.7, \\ -10f_{evol} + 8, & \text{if } 0.7 < f_{evol} \leq 0.8, \\ 0, & \text{if } 0.8 < f_{evol} \leq 1.0. \end{cases} \tag{3.8}$$

The algorithm stops the execution of the ABC only when the evolutionary state is *Jumping out* or *Exploration*. During the execution of the ABC, the memory of the particles is stored. Before the restart of the APSO, it is necessary to compare the memory of the particles and the food sources. If the food source of the bee is better than the position stored in the memory, then the memory of the particle is updated, otherwise the memory is reestablished.

3.2.4 Stagnation Counter

This parameter works similarly the stagnation counter of ABC algorithm. It is a improvement of algorithm, because we observed that the ABeePSO algorithm lost the exploitation capability. The proposal uses like condition to execution of operator based on artificial bees the evolutionary state, *i.e.*, when is *Convergence* state, execute the modified ABC. Thus, the algorithm do not allow the deep exploration of the search region, because as from this moment the swarm expands.

Thus, the stagnation counter allows that swarm explore the search region during more iterations. Each particle has a stagnation counter. If the particle does not improve its position, this counter is increased. The combination of the stagnation counter with the evolutionary state guarantees a better exploitation ability.

3.2.5 Pseudocode of the ABeePSO Algorithm

The pseudocode of the ABeePSO algorithm is shown in Algorithm 7. In the pseudocode, we observe the same steps of APSO algorithm, but the lines 12 to 26 show the modified ABC. In the line 14 the guide bees movement is performed and the line 18 presents the guided bees movement.

3.3 Clan Adaptive Bee and Particle Swarm Optimization

The Clan Adaptive Bee and Particle Swarm Optimization (ClanABeePSO) is an algorithm that presents the characteristics of our proposed algorithm with communication topology of clans, found in the ClanPSO algorithm. Similar to the ClanAPSO algorithm, the use of the cooperative behavior with multi swarm system allows that the sub-swarms can perform different types of search by iteration of the algorithm. It is possible to observe different types of agents (particles and bees) as well.

As the ClanPSO algorithm, each clan use a fully-connected structure. However, for each iteration, each one of clans performs a search using the ABeePSO algorithm and selects the particle with the best information of the entire clan (delegation of leader). The leaders adjust their positions running a APSO or PSO execution and the conference can be performed using either global or local topology.

3.3.1 Pseudocode of the ClanABeePSO Algorithm

The simplified pseudocode of the ClanABeePSO algorithm is shown in Algorithm 8. In lines 3 to 6 have the ABeePSO execution within each clan. But, the other parts of pseudocode are similar to the ones presented in the ClanAPSO algorithm.

Algorithm 7: Pseudocode of the ABeePSO algorithm.

```
1  Initialize particles of swarm;
2  while the stop criterion is not achieved do
3      for each particle do
4          Evaluate the average distance according to Equation (2.4);
5      end
6      Calculate the evolutionary factor according to Equation (2.5);
7      Calculate the membership function according to Equations (3.6), (3.7),
       (3.8) and (2.9);
8      Classify the swarm in the evolutionary state;
9      Adapt the acceleration coefficients according to Table 2.1;
10     Normalize the acceleration coefficients according to Equation (2.10);
11     Update the inertial factor according to Equation (2.13);
12     if classified on Convergence state then
13         while Does not classified on Exploration or Jumping out state do
14             Execute guide bees movement according to Equation (3.1);
15             for each guide bee do
16                 Calculate the probability od roulette wheel selection using
                   the Equation (2.16);
17             end
18             Execute guided bees movement according to Equation (3.5);
19             for each guide bee do
20                 Calculate the average distance according to Equation (2.4);
21             end
22             Calculate the evolutionary factor according to Equation (2.5);
23             Calculate the membership function according to Equations (3.6),
               (3.7), (3.8) and (2.9);
24             Classify the swarm in the evolutionary state;
25         end
26     end
27     Update the memory of particles using food sources;
28     for each particle do
29         Update the social memory in neighborhood;
30         Update the velocity according to Equation (2.1);
31         Update the position according to Equation (2.2);
32         Evaluate the position using the objective function of problem;
33         if current position is better than cognitive memory then
34             Update the cognitive memory;
35         end
36         if current position is better than social memory then
37             Update the social memory;
38         end
39     end
40 end
41 return the social memory.
```

Algorithm 8: Pseudocode of ClanABeePSO algorithm.

```
1  Initialize the particles with random positions and velocities;
2  while the stop criterion is not achieved do
3      for each clan do
4          Execute ABeePSO within the clans;
5          Delegate leader;
6      end
7      Create conference of leaders;
8      Execute APSO (or PSO) with the leaders;
9  end
10 return the best particle.
```

Chapter 4

Methodology and Experiments Setup

This chapter presents the benchmark minimization problems used in our experiments and the experimental methodology, including the test environment, evaluation metrics and experimental setup.

4.1 Benchmark Functions

We use the benchmark functions proposed by Tang *et al.* in 2010 [TLS+10], that are appropriate to optimization of high dimensionality. There are 20 functions, which have different degrees of separability and are divided in four types of high-dimensional problems [TLS+10]: (i) Separable functions; (ii) Partially-separable functions, in which a small number of variables are dependent while all the remaining ones are independent; (iii) Partially-separable functions that consist of multiple independent subcomponents, each of which is m-non-separable; and (iv) Fully-nonseparable functions.

The benchmark functions use the following functions as basic functions [TLS+10]:

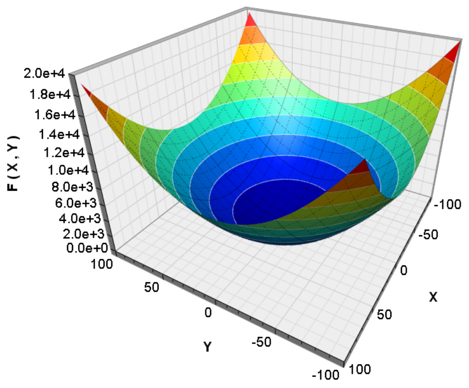
- The Sphere Function: The Sphere function (observe Figure 4.1) is defined according to Equation (4.1).

$$F_{Sphere}(\vec{x}) = \sum_{i=1}^{D} x_i^2, \tag{4.1}$$

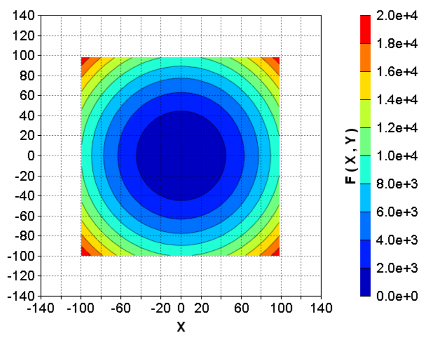
  in which D is the number of dimensions and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix). The Sphere function is very simple and is manly used for optimization.

- The Rotated Elliptic Function: the original Elliptic Function is separable, and is defined according to Equation (4.2).

$$F_{Elliptic}(\vec{x}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} x_i^2, \tag{4.2}$$



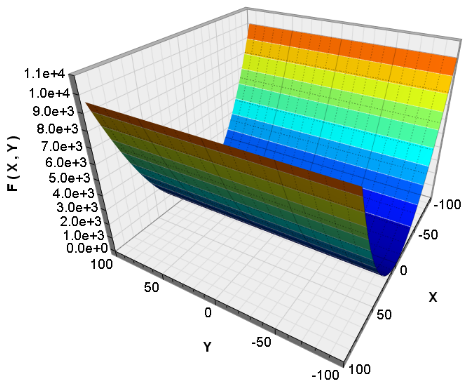
(a)



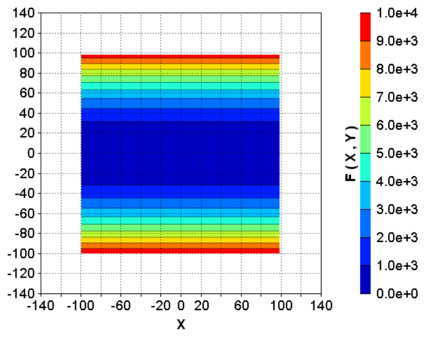
(b)

Figure 4.1: Illustration of Sphere function: (a) Surface and (b) Contour.

in which D is the number of dimensions and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix). The number 10^6 is called condition number, which is used to transform a Sphere function to an Elliptic function [SNHD+05].



(a)



(b)

Figure 4.2: Illustration of Elliptic function: (a) Surface and (b) Contour.

To make this function be nonseparable, an orthogonal matrix will be used to rotate the coordinates. The rotated Elliptic function is defined according to Equation 4.3.

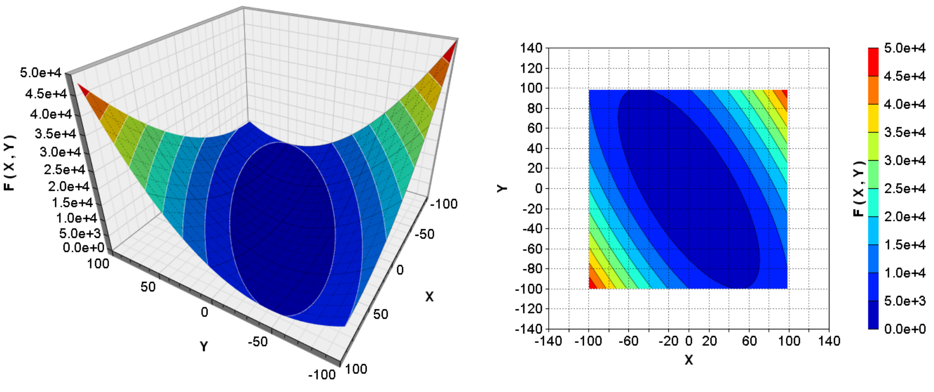
$$F_{Rot\_Elliptic}(\vec{x}) = F_{Elliptic}(\vec{z}), \quad \vec{z} = \vec{x} * \vec{M}, \tag{4.3}$$

in which $\vec{M}$ is a $D \times D$ orthogonal matrix, and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix).

- Schwefel's Problem 1.2: is a naturally nonseparable, which is defined according to Equation (4.4). Figure 4.3 illustrates Schwefel's Problem 1.2.

$$F_{Schwefel}(\vec{x}) = \sum_{i=1}^{D}\Big(\sum_{j=1}^{i} x_i\Big)^2, \tag{4.4}$$

in which D is the number of dimensions and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix).



(a) (b)

Figure 4.3: Illustration of Schwefel's Problem 1.2: (a) Surface and (b) Contour.

- Rosenbrock's Function: Figure 4.4 presents an illustration of Rosenbrock's function, that is also naturally nonseparable and is defined according to Equation (4.5).

$$F_{Rosenbrock}(\vec{x}) = \sum_{i=1}^{D-1}[100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2], \tag{4.5}$$

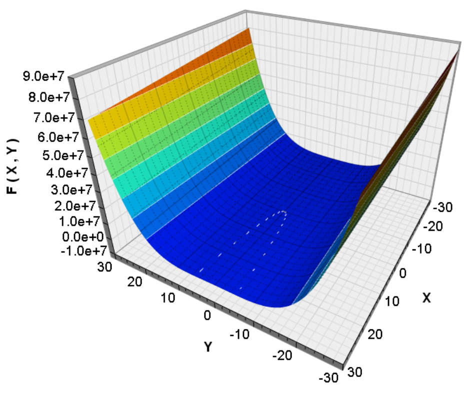
in which $D \geq 2$ is the number of dimensions and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix).

- The Rotated Rastrigin's Function: The original Rastrigin's function, that can be visualized in Figure 4.5, is separable, and is defined according to Equation (4.6).

$$F_{Rastrigin}(\vec{x}) = \sum_{i=1}^{D}[x_i^2 - 10cos(2\pi x_i) + 10], \tag{4.6}$$

in which D is the number of dimensions and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix). Similarly, to make it nonseparable, an orthogonal matrix is also used for coordinate rotation. The Rotated Rasntrigin's function is defined according to Equation (4.7).

$$F_{Rot\_Rastrigin}(\vec{x}) = F_{Rastrigin}(\vec{z}), \quad \vec{z} = \vec{x} * \vec{M}, \tag{4.7}$$


(a)

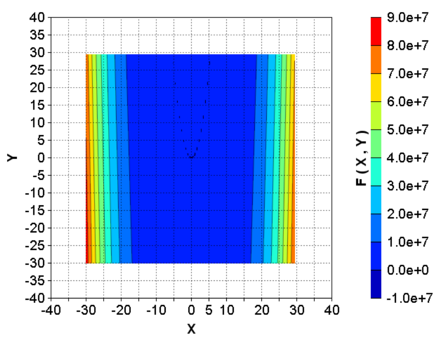
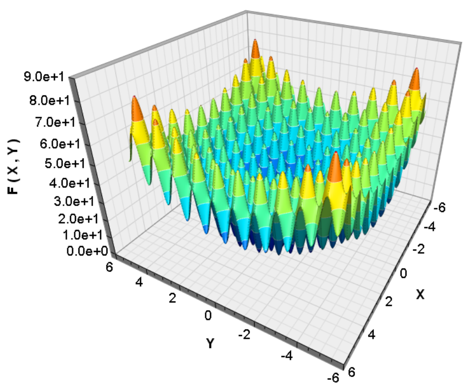

(b)

Figure 4.4: Illustration of Rosenbrock's function: (a) Surface and (b) Contour.


(a)

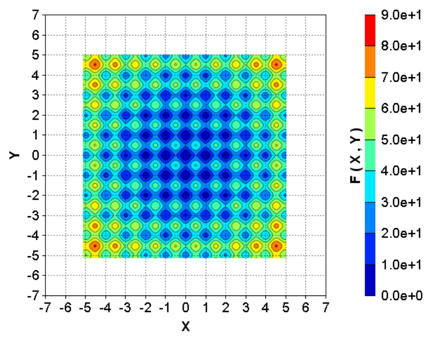

(b)

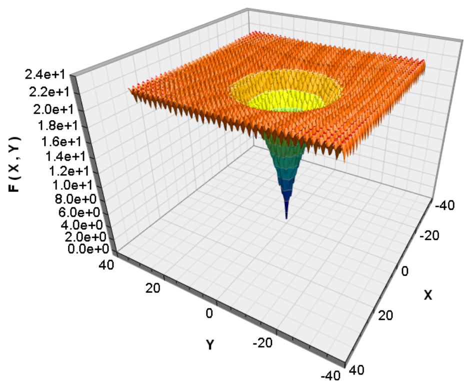
Figure 4.5: Illustration of Rastrigin's function: (a) Surface and (b) Contour.

in which $\vec{M}$ is a $D \times D$ orthogonal matrix, and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix). Rastrigin's function is a classical multimodal problem. It is difficult since the number of local optima grows exponentially with the increase of dimensionality.

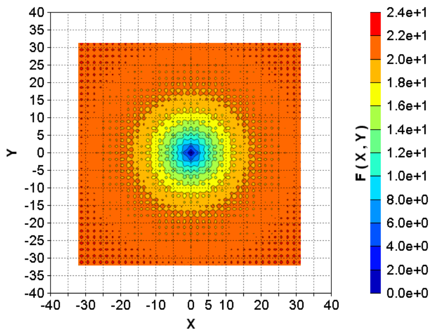
- The Rotated Ackley's Function: The original Ackley's function is separable, and is defined according to Equation (4.8) and the graphic is illustrated in Figure 4.6.

$$F_{Ackley}(\vec{x}) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e, \tag{4.8}$$

in which D is the number of dimensions and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix). To make it nonseparable, an orthogonal matrix is used again for coordinate rotation. The Rotated



(a)



(b)

Figure 4.6: Illustration of Ackley's function: (a) Surface and (b) Contour.

Ackley's function is defined according to Equation (4.9).

$$F_{Rot\_Ackley}(\vec{x}) = F_{Ackley}(\vec{z}), \quad \vec{z} = \vec{x} * \vec{M}, \tag{4.9}$$

in which $\vec{M}$ is a $D \times D$ orthogonal matrix, and $\vec{x} = (x_1, x_2, \ldots, x_D)$ is a D-dimensional row vector (i.e., a $1 \times D$ matrix).

The benchmark functions are listed according to the separability degree of variables. The following, we describe the mathematical formulas and properties of these functions.

4.1.1 Separable Functions

In this subsection, the variables are: (i) $D = 1000$ is the number of dimensions; (ii) $\vec{x} = (x_1, x_2, \ldots, x_D)$ is the candidate solution; and (iii) $\vec{z} = (z_1, z_2, \ldots, z_D)$ is the shifted candidate solution, where $\vec{z} = \vec{x} - \vec{o}$, $\vec{o} = (o_1, o_2, \ldots, o_D)$ is the (shifted global optimum). The functions of this group are listed as follows.

- $F1$ - Shifted Elliptic Function: the Shifted Elliptic function is unimodal, separable, scalable and its formula is defined according to Equation (4.10).

$$F_1(\vec{x}) = F_{Elliptic}(\vec{z}) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2; \tag{4.10}$$

- $F2$ - Shifted Rastrigin's Function: the Shifted Rastrigin's function is multimodal, separable, scalable and its formula is defined according to Equation (4.11).

$$F_2(\vec{x}) = F_{Rastrigin}(\vec{z}) = \sum_{i=1}^{D} [z_i^2 - 10cos(2\pi z_i) + 10]; \tag{4.11}$$

- $F3$ - Shifted Ackley's Function: the Shifted Ackley's function is multimodal, separable, scalable and its formula is defined according to Equation (4.12).

$$F_3(\vec{x}) = F_{Ackley}(\vec{z}) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi z_i)\right) + 20 + e. \tag{4.12}$$

4.1.2 Single-group m-nonseparable Functions

In this subsection, the variables are: (i) $D = 1000$ is the number of dimensions; (ii) $m = 50$ is group size, (iii) P is a random permutation of $1, 2, \ldots, D$; (iv) $\vec{x} = (x_1, x_2, \ldots, x_D)$ is the candidate solution; and (v) $\vec{z} = (z_1, z_2, \ldots, z_D)$ is the shifted candidate solution, where $\vec{z} = \vec{x} - \vec{o}$, $\vec{o} = (o_1, o_2, \ldots, o_D)$ is the (shifted global optimum). The functions of this group are listed as follows.

- $F4$ - Single-group Shifted and m-rotated Elliptic Function: the Single-group Shifted and m-rotated Elliptic function is unimodal, has single-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.13).

$$F_4(\vec{x}) = F_{Rot\_Elliptic}[\vec{z}(P_1 : P_m)] * 10^6 + F_{Elliptic}[\vec{z}(P_{m+1} : P_D)], \tag{4.13}$$

- $F5$ - Single-group Shifted and m-rotated Rastrigin's Function: the Single-group Shifted and m-rotated Rastrigin's function is multimodal, has single-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.14).

$$F_5(\vec{x}) = F_{Rot\_Rastrigin}[\vec{z}(P_1 : P_m)] * 10^6 + F_{Rastrigin}[\vec{z}(P_{m+1} : P_D)], \tag{4.14}$$

- $F6$ - Single-group Shifted and m-rotated Ackley's Function: the Single-group Shifted and m-rotated Ackley's function is multimodal, has single-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.15).

$$F_6(\vec{x}) = F_{Rot\_Ackley}[\vec{z}(P_1 : P_m)] * 10^6 + F_{Ackley}[\vec{z}(P_{m+1} : P_D)], \tag{4.15}$$

- $F7$ - Single-group Shifted and m-dimensional Schwefel's Problem 1.2: the Single-group Shifted and m-dimensional Schwefel's Problem 1.2 is unimodal, has single-group with m nonseparable variables, and its formula is defined according to Equation (4.16).

$$F_7(\vec{x}) = F_{Schwefel}[\vec{z}(P_1 : P_m)] * 10^6 + F_{Sphere}[\vec{z}(P_{m+1} : P_D)], \tag{4.16}$$

- $F8$ - Single-group Shifted and m-dimensional Rosenbrock's Function: The Single-group Shifted and m-dimensional Rosenbrock's function is multimodal, has single-group with m nonseparable variables, and its formula is defined according to Equation (4.17).

$$F_8(\vec{x}) = F_{Rosenbrock}[\vec{z}(P_1 : P_m)] * 10^6 + F_{Sphere}[\vec{z}(P_{m+1} : P_D)], \tag{4.17}$$

4.1.3 $D/2m$-group m-nonseparable Functions

In this subsection, the variables are: (i) $D = 1000$ is the number of dimensions; (ii) $m = 50$ is group size; (iii) P is a random permutation of $1, 2, \ldots, D$, (iv) $(iv)\vec{x} = (x_1, x_2, \ldots, x_D)$ is the candidate solution; and (v) $\vec{z} = (z_1, z_2, \ldots, z_D)$ is the shifted candidate solution, where $\vec{z} = \vec{x} - \vec{o}$, $\vec{o} = (o_1, o_2, \ldots, o_D)$ is the (shifted global optimum). The functions of this group are listed as follows.

- $F9$ - $D/2m$-group Shifted and m-rotated Elliptic Function: The $D/2m$-group Shifted and m-rotated Elliptic function is unimodal, has $D/2m$-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.18).

$$F_9(\vec{x}) = \sum_{k=1}^{\frac{D}{2m}} F_{Rot\_Elliptic}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})] + F_{Elliptic}[\vec{z}(P_{\frac{D}{2}+1} : P_D)], \tag{4.18}$$

- $F10$ - $D/2m$-group Shifted and m-rotated Rastrigin's Function: The $D/2m$-group Shifted and m-rotated Rastrigin's function is multimodal, has $D/2m$-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.19).

$$F_{10}(\vec{x}) = \sum_{k=1}^{\frac{D}{2m}} F_{Rot\_Rastrigin}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})] + F_{Rastrigin}[\vec{z}(P_{\frac{D}{2}+1} : P_D)], \tag{4.19}$$

- $F11$ - $D/2m$-group Shifted and m-rotated Ackley's Function: The $D/2m$-group Shifted and m-rotated Ackley's function is multimodal, has $D/2m$-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.20).

$$F_{11}(\vec{x}) = \sum_{k=1}^{\frac{D}{2m}} F_{Rot\_Ackley}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})] + F_{Ackley}[\vec{z}(P_{\frac{D}{2}+1} : P_D)], \tag{4.20}$$

- $F12$ - $D/2m$-group Shifted and m-dimensional Schwefel's Problem 1.2: The $D/2m$-group Shifted and m-dimensional Schwefel's Problem 1.2 is unimodal, has $D/2m$-group with m nonseparable variables, and its formula is defined according to Equation (4.21).

$$F_{12}(\vec{x}) = \sum_{k=1}^{\frac{D}{2m}} F_{Schwefel}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})] + F_{Sphere}[\vec{z}(P_{\frac{D}{2}+1} : P_D)], \tag{4.21}$$

- $F13$ - $D/2m$-group Shifted and m-dimensional Rosenbrock's Function: The $D/2m$-group Shifted and m-dimensional Rosenbrock's function is multimodal, has $D/2m$-group with m nonseparable variables, and its formula is defined according to Equation (4.22).

$$F_{13}(\vec{x}) = \sum_{k=1}^{\frac{D}{2m}} F_{Rosenbrock}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})] + F_{Sphere}[\vec{z}(P_{\frac{D}{2}+1} : P_D)], \tag{4.22}$$

4.1.4 D/m-group m-nonseparable Functions

In this subsection, the variables are: (i) $D = 1000$ is the number of dimensions; (ii) $m = 50$ is group size; (iii) P is a random permutation of $1, 2, \ldots, D$; (iv) $\vec{x} = (x_1, x_2, \ldots, x_D)$ is the candidate solution; and (v) $\vec{z} = (z_1, z_2, \ldots, z_D)$ is the shifted candidate solution, where $\vec{z} = \vec{x} - \vec{o}$, $\vec{o} = (o_1, o_2, \ldots, o_D)$ is the (shifted global optimum). The functions of this group are listed as follows.

- $F14$ - D/m-group Shifted and m-rotated Elliptic Function: The D/m-group Shifted and m-rotated Elliptic function is unimodal, has $D/2m$-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.23).

$$F_{14}(\vec{x}) = \sum_{k=1}^{\frac{D}{m}} F_{Rot\_Elliptic}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})], \tag{4.23}$$

- $F15$ - D/m-group Shifted and m-rotated Rastrigin's Function: The D/m-group Shifted and m-rotated Rastrigin's function is multimodal, has D/m-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.24).

$$F_{15}(\vec{x}) = \sum_{k=1}^{\frac{D}{m}} F_{Rot\_Rastrigin}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})], \tag{4.24}$$

- $F16$ - D/m-group Shifted and m-rotated Ackley's Function: The D/m-group Shifted and m-rotated Ackley's function is multimodal, has D/m-group with m rotated and nonseparable variables, and its formula is defined according to Equation (4.25).

$$F_{16}(\vec{x}) = \sum_{k=1}^{\frac{D}{m}} F_{Rot\_Ackley}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})], \tag{4.25}$$

- $F17$ - D/m-group Shifted and m-dimensional Schwefel's Problem 1.2: The D/m-group Shifted and m-dimensional Schwefel's Problem 1.2 is unimodal, has D/m-group with m nonseparable variables, and its formula is defined according to Equation (4.26).

$$F_{17}(\vec{x}) = \sum_{k=1}^{\frac{D}{m}} F_{Schwefel}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})]; \tag{4.26}$$

- $F18$ - D/m-group Shifted and m-dimensional Rosenbrock's Function: The D/m-group Shifted and m-dimensional Rosenbrock's function is multimodal, has D/m-group with m nonseparable variables, and its formula is defined according to Equation (4.27).

$$F_{18}(\vec{x}) = \sum_{k=1}^{\frac{D}{m}} F_{Rosenbrock}[\vec{z}(P_{(k-1)*m+1} : P_{k*m})]. \tag{4.27}$$

4.1.5 Nonseparable Functions

In this subsection, the variables are: (i) $D = 1000$ is the number of dimensions; (ii) $\vec{x} = (x_1, x_2, \ldots, x_D)$ is the candidate solution; and (iii) $\vec{z} = (z_1, z_2, \ldots, z_D)$ is the shifted candidate solution, where $\vec{z} = \vec{x} - \vec{o}$, $\vec{o} = (o_1, o_2, \ldots, o_D)$ is the (shifted global optimum). The functions of this group are listed as follows.

- $F19$ - Shifted Schwefel's Problem 1.2: The Shifted Schwefel's Problem 1.2 is unimodal, fully-nonseparable and its formula is defined according to Equation (4.28).

$$F_{19}(\vec{x}) = F_{Schwefel}(\vec{z}) \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_i \right)^2; \tag{4.28}$$

- $F20$ - Shifted Rosenbrock's Function: The Shifted Rosenbrock's function is multimodal, fully-nonseparable and its formula is defined according to Equation (4.29).

$$F_{20}(\vec{x}) = F_{Rosenbrock}(\vec{z}) = \sum_{i=1}^{D-1} [100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2]. \tag{4.29}$$

4.1.6 Definition of Search Space Boundaries and Global Optimum

The Table 4.1 presents the search space boundaries and global optimum of each benchmark function. As observed, these functions are minimization problems.

Table 4.1: Benchmark functions, search space boundaries and global optimum position.

Function	Search Space	Global Optimum
$F1$	$-100 \leq x_i \leq 100$	0.0^D
$F2$	$-5 \leq x_i \leq 5$	0.0^D
$F3$	$-32 \leq x_i \leq 32$	0.0^D
$F4$	$-100 \leq x_i \leq 100$	0.0^D
$F5$	$-5 \leq x_i \leq 5$	0.0^D
$F6$	$-32 \leq x_i \leq 32$	0.0^D
$F7$	$-100 \leq x_i \leq 100$	0.0^D
$F8$	$-100 \leq x_i \leq 100$	0.0^D
$F9$	$-100 \leq x_i \leq 100$	0.0^D
$F10$	$-5 \leq x_i \leq 5$	0.0^D
$F11$	$-32 \leq x_i \leq 32$	0.0^D
$F12$	$-100 \leq x_i \leq 100$	0.0^D
$F13$	$-100 \leq x_i \leq 100$	0.0^D
$F14$	$-100 \leq x_i \leq 100$	0.0^D
$F15$	$-5 \leq x_i \leq 5$	0.0^D
$F16$	$-32 \leq x_i \leq 32$	0.0^D
$F17$	$-100 \leq x_i \leq 100$	0.0^D
$F18$	$-100 \leq x_i \leq 100$	0.0^D
$F19$	$-100 \leq x_i \leq 100$	0.0^D
$F20$	$-100 \leq x_i \leq 100$	0.0^D

4.2 Experimental Methodology

This section presents the strategies adopted in the realization the experiments showed in this dissertation. The topics of this section are: environment of simulation, in which the framework used to realized the experiments is described, evaluation metrics used to assess the performance of the algorithms, and the simulation setup, in which the values of the parameters are presented.

4.2.1 Environment of the Simulations

All simulations were realized in a framework, called *Dynamic Optimization System Simulator* (DOSS) proposed by Carneiro [dSC10]. This framework presents a suitable infrastructure to develop and simulate the algorithms. It was developed using the Java programming language [AGH05], the Eclipse tool [Fou11b] and also Maven tool [Fou11a]. In the Eclipse tool, we can write the source code and the Maven tool is used to manage the dependencies and compile the projects. The framework has the graphical user interface (GUI), called *Dynamic Optimization System Analyzer* (DOSA). This interface allows one to create test scenarios and evaluate the simulations results [dSC10].

Figure 4.7 illustrates the main screen of DOSA, in which the test scenarios are build selecting: the algorithm, the optimization problem, the evaluation metric, and the stopping criteria. Then, one can define the configuration of the experiment, assigning values for the parameters of the algorithm, the number of dimensions of the problem, the stop condition and the number of trials.

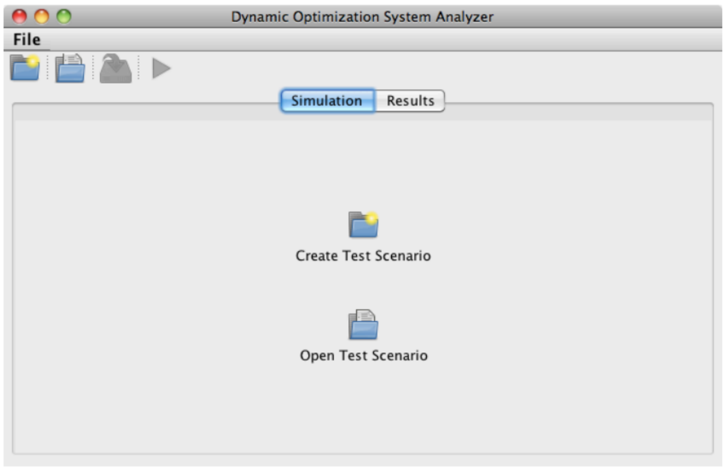


Figure 4.7: Illustration of the DOSA main screen.

We used the software called OriginPro version 8.6 to build the graphics [Cor11] of generated results. OriginPro is proprietary software very used to generate interactive scientific graphing and data analysis.

4.2.2 Metrics of Evaluation

The evaluation metrics used to compare the performance of the algorithms are: the convergence velocity, it measures how fast the algorithm converges; the scalability, it estimates the degradation of the performance as the complexity of the problem (number of dimensions) increases; and the average value for the fitness at the end of each trial. We also performed statistical tests when the difference between the results of the algorithms assessing the boxplots is not too large. We used the Wilcoxon non-parametric test [Low11].

4.2.3 Experiments Setup

In all experiments, we used 30 particles to ABeePSO, APSO and PSO and 60 bees to ABC. We used a double of bees in ABC because just a half are food sources and we would like to assess the same number of potential solutions. We used 6 clans for the algorithms ClanPSO, ClanAPSO and ClanABeePSO, *i.e.* the populations of these algorithms are composed by 6 clans of 5 particles. We used the local communication topology for the ABeePSO, APSO and PSO algorithm. The stagnation counter ($MaxTrial$) of ABC algorithm is equal to 5.

We used the number of dimensions equal to 100, 300, 500 or 1,000 dimensions. The number of iterations depends on the number of dimensions, we performed

1,000, 3,000, 5,000 and 10,000 iterations, respectively. The stop criterion for these algorithms is the number of iterations. This condition was chosen because it limits the number of fitness evaluations and allows a fair comparison between the algorithms.

Table 4.2: The number of parameters of each algorithm.

Algorithm	**Number of Parameters**	**Parameters**
PSO	5	N, c_1, c_2, ω_{min}, ω_{max}
APSO	5	N, c_1, c_2, ω_{min}, ω_{max}
ClanPSO	6	N, c_1, c_2, ω_{min}, ω_{max}, N_{pc}
ClanAPSO	6	N, c_1, c_2, ω_{min}, ω_{max}, N_{pc}
ABC	2	$2 \cdot SN$, $MaxTrial$
FSS	6	S, W_{scale}, $step_{vol\_initial}$, $step_{vol\_final}$, $step_{ind\_initial}$, $step_{ind\_final}$

Chapter 5

Simulation Results

This chapter presents the results achieved by the algorithms depicted in the Chapters 2 and 3 using all optimization problems described in the Chapter 4. This Chapter is divided in three parts:

- A parametric analysis of the ABeePSO:
  - Analysis of dispersion step;
  - Analysis of acceleration coefficients (c_1 and c_2);
  - Analysis of intervals of fuzzy membership functions;
  - Analysis of stagnation counter;
- Performance comparison among ABeePSO, PSO, APSO and ABC:
  - Analysis of convergence;
  - Analysis of the dimensionality;
- Initial analysis of the ClanABeePSO algorithm.

5.1 Parametric Analysis of the ABeePSO

This section presents the parametric analysis for the ABeePSO algorithm. This analysis aims to adapt the new operator and avoid explosion states in the ABeePSO algorithm. In this analysis, the number of dimensions is equal to 100 and the number of iterations is 1,000. All results were obtained over 10 trials.

5.1.1 Analysis of Dispersion Step

This analysis aims to investigate the influence of the parameters α and β used to calculate the dispersion step of the guide bees (observe the Equation (3.3)). We use one unimodal function ($F1$) and one multimodal function ($F16$). We selected a logistic function since we aim to have a step value around 1.0 for low f_{evol} values and low step values for high f_{evol} values.

We varied the value of α between 0 and 100, for $\beta = 0.1$, 0.2 and 0.5. According to Figure 5.1, we observed that the best results were obtained for $\beta = 0.5$. We also observed that the variation of α does not significantly interfere in the performance for the studied unimodal functions ($F1$) (Figure 5.1(a)). In the multimodal function ($F16$), we observed that lower α values led to better results (Figure 5.1(b)). Because of this, we assume $\alpha = 10.0$ and $\beta = 0.5$ for further simulations.

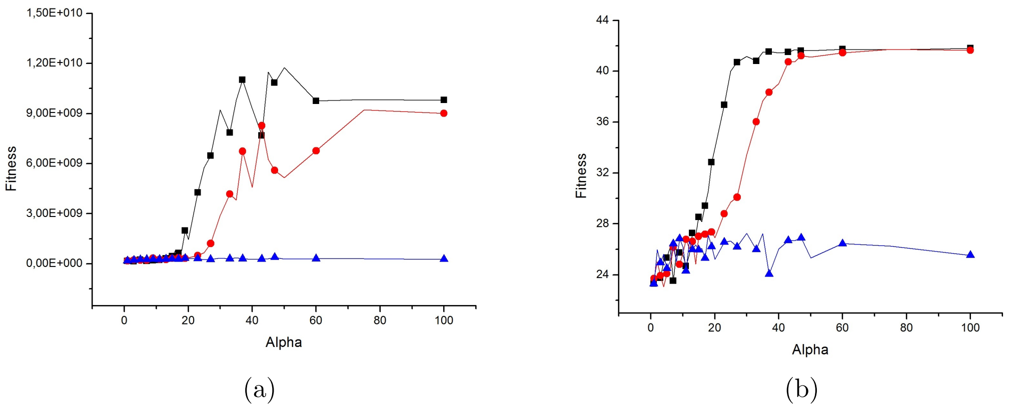


Figure 5.1: Performance of the ABeePSO algorithm as a function of α for (a) $F1$ and (b) $F16$, considering $\beta = 0.1$ (squares), $\beta = 0.2$ (circles) and $\beta = 0.5$ (triangle).

5.1.2 Analysis of Acceleration Coefficients (c_1 and c_2)

Since the acceleration coefficients (c_1 and c_2) must be updated at each iteration and our proposal presents a different dynamic behavior, it is necessary to analyze the impact of the variation rate of c_1 and c_2 on the performance of our proposal. δ defines the increment (or decrement) of c_1 and c_2 per iteration. As lightly increment (or decrement) is performed by using $\delta/2$.

Table 5.1 shows the average fitness value for the functions $F4$ (unimodal) and $F13$ (multimodal) for different policies to determine δ. The influence on the performance is not significant for the unimodal function, but the best result was achieved for a $\delta = 0.2$. For the multimodal ($F13$), we the best results where achieved when δ is randomly chosen from the intervals $[0.01; 0.05]$ and $[0.05; 0.1]$. Since the difference between the results is not significant, we chose the same interval defined in the original APSO for further simulations, *i.e.* δ randomly chosen from the interval $[0.05; 0.1]$.

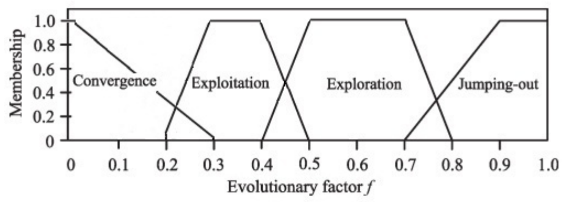
5.1.3 Analysis of Intervals of Fuzzy Membership Functions

This analysis is important since it determines the shift between APSO and ABeePSO. We performed some simulations for the functions $F4$ (unimodal) and $F13$ (multimodal) for different Fuzzy membership functions. The configuration 1 uses the Fuzzy membership functions depicted in Figure 3.2. In the configuration 2 we

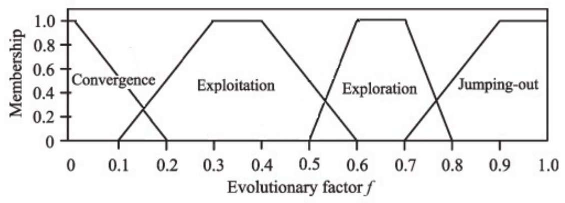
Table 5.1: Impact of variation of acceleration rate in quality of search.

Value of δ	$F4$	$F13$
Random [0.01;0.1]	1.23E+14	1.02E+08
Random [0.01;0.05]	1.33E+14	**5.26E+07**
Random [0.02;0.1]	1.21E+14	5.91E+07
Random [0.02;0.2]	1.38E+14	8.42E+07
Random [0.05;0.1]	1.32E+14	5.72E+07
Random [0.05;0.2]	1.37E+14	5.82E+07
Fixed at 0.1	1.30E+14	8.91E+07
Fixed at 0.2	**1.13E+14**	6.56E+07
Fixed at 0.05	1.32E+14	1.05E+08

modified the interval of the *Convergence* state from 0.2 to 0.3 and the threshold of the *Exploitation* state from 0.1 to 0.2, as one can observe in Figure 5.2(a). This variation allows the algorithm to anticipate the start of the execution of the ABC based operator. In the configuration 3 we modified the interval of *Exploitation* state from 0.5 to 0.6 and the threshold of the *Exploration* state from 0.4 to 0.5, one can observe Figure 5.2(b). This variation allows the execution of the ABC based operator for a longer time.



(a)



(b)

Figure 5.2: Different configurations to analysis of intervals of fuzzy membership functions: (a) Configuration 2 and (b) Configuration 3.

For the unimodal function ($F7$), the configurations 1, 2 and 3 achieved average fitness (standard deviation) equal to $4.2E+10(8.53E+09)$, $4.59E+10(5.43E+09)$ and $4.9E+10(1.31E+09)$, respectively. For the multimodal function ($F20$), the configurations 1, 2 and 3 achieved the following average fitness (standard deviation) equal to $8.23E+08(2.5E+08)$, $1.02E+09(6.39E+08)$ and $1.29E+09(4.6E+08)$, respectively. The results indicate the configuration depicted in Figure 3.2, *i.e.* configuration 1. In other benchmark functions we observed that the dependence on the fuzzy intervals is higher for the multimodal functions, but the configuration depicted in Figure 3.2 is still the best option.

5.1.4 Analysis of Stagnation Counter

This initial analysis aims to estimate the initial value for stagnation counter and evaluate the behavior of algorithm with this information. We used one unimodal functions ($F4$) and multimodal function ($F2$).

We varied the value of stagnation counter is ranging between 0 and 100. According to results, we observed a similar performance for both functions. For the multimodal function, when the stagnation counter value is medium or high, the fitness value does not improve, *i.e.* it is necessary to expand the swarm by using the ABC based operator and generate diversity. For the unimodal function, the best results were obtained when the stagnation counter value is low or medium. When the value is very high, the obtained fitness is mitigated. From this analysis, we defined a initial value equal 10.0 for stagnation step.

5.2 Performance comparison among ABeePSO, PSO, APSO and ABC

This section presents a performance analysis of the ABeePSO algorithm and a comparison to the PSO, APSO and ABC algorithms. We assessed the convergence velocity and scalability. In the analysis of convergence, the number of dimensions is equal to 100 and the number of iterations is equal to 1,000. In the analysis of dimensionality, we performed 3,000, 5,000 and 10,000 iterations for 300, 500 and 1,000 dimensions, respectively. All results were obtained over 50 trials.

5.2.1 Analysis of Convergence

We compared the performance of various algorithms along the iterations to analyze the convergence velocity. Figure 5.3 depicts the evolution of ABC, APSO PSO and ABeePSO algorithms for multimodal and unimodal benchmark functions. The Figures 5.3(a) and 5.3(b) are multimodal functions F3 and F5, respectively and Figures 5.3(c) and 5.3(d) are unimodal functions F12 and F14, respectively. We did not observe a significant improvement in the convergence velocity for unimodal function. However, our proposal outperformed the other approaches for the multimodal problems, mainly for the F3 function.

5.2.2 Analysis of Scalability

We also analyzed the performance of the algorithms as a function of the number of dimensions of the problem. We aim to analyze the scalability of the algorithms. As the dimensionality increases, the difficulty of the problems also increases. Figures

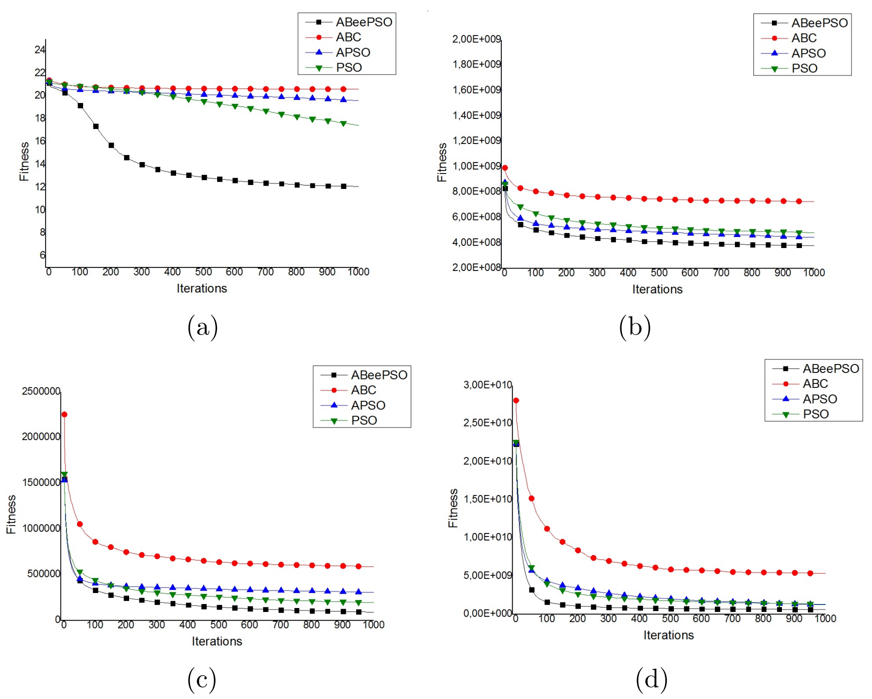


Figure 5.3: Analysis of Convergence: Fitness function evolution of the ABC, APSO, PSO and ABeePSO algorithms for the $D = 100$ in (a) $F3$, (b) $F5$, (c) $F12$ and (d) $F14$ functions.

5.4 and 5.5 show the boxplot of the fitness as a function of the dimensionality for the benchmark functions $F7$ (unimodal) and $F11$ (multimodal), respectively.

For the unimodal function, one can observe that the ratio between the performance of the algorithms is maintained, but it is smaller for $D = 1{,}000$.

For the multimodal function, we obtained better results when compared to other approaches, *i.e.* the difference in the performance between our proposal and other techniques is maintained as the dimensionality of the problem also increases.

We realized experiments with all benchmark functions, varying the number of dimensions. We made the comparison the ABeePSO with ABC, APSO and PSO algorithms. Tables 5.2, 5.3, 5.4 and 5.5 present the average fitness and (standard deviation) obtained from simulations for 100, 300, 500 and 1,000 dimensions, respectively. The best average fitness values is highlighted for each benchmark function.

Since the difference between the algorithm is not large in some few cases, we performed a comparison by using a statistical test. We show the results for the Wilcoxon Test with significance level of 0.05. Up-triangle means that our approach is better, Down-triangle means that our approach is worst and "-" means that there is no statistical difference. Tables 5.6, 5.7, 5.8 and 5.9 present the average fitness and (standard deviation) obtained from simulations for 100, 300, 500 and 1,000 dimensions, respectively. The test shows that our proposal is superior in most of cases, except for $F2$ and $F10$ in 100 dimensions, $F2$, $F15$ and $F19$ in 300 dimensions and $F2$, $F9$, $F10$, $F12$, $F14$, $F15$ and $F19$ in 500 dimensions. Our

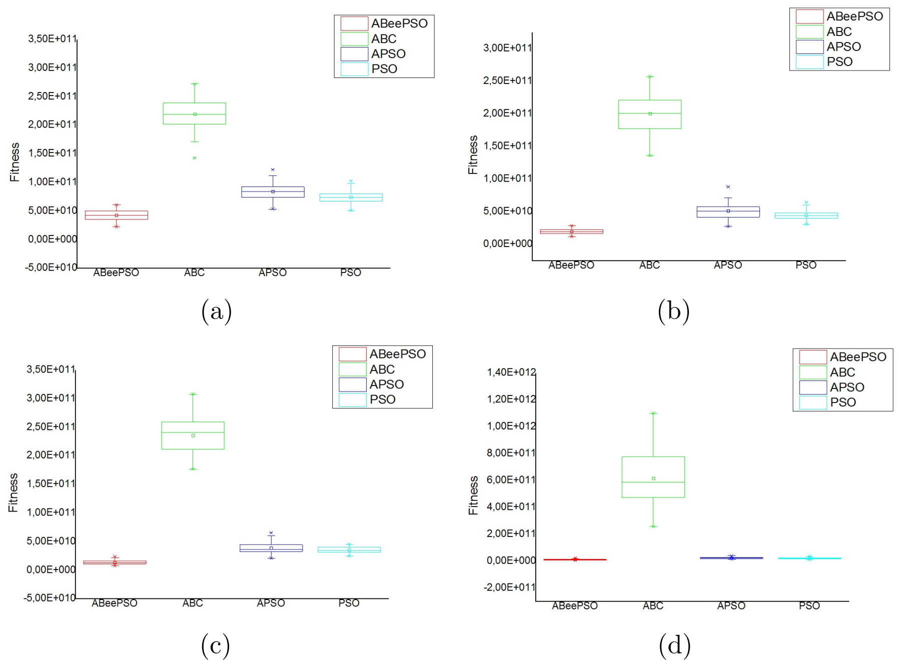


Figure 5.4: Analysis of the Dimensionality: Boxplot of the fitness for the $F7$ function varying the number of dimensions for (a) $D = 100$, (b) $D = 300$, (c) $D = 500$ and (d) $D = 1{,}000$.

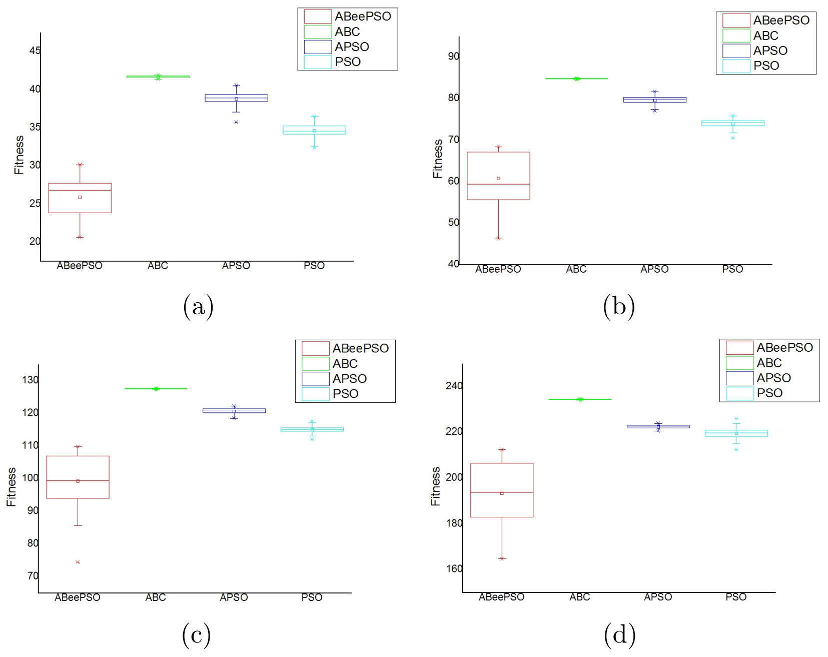


Figure 5.5: Analysis of the Dimensionality: Boxplot of the fitness for the $F11$ function varying the number of dimensions for (a) $D = 100$, (b) $D = 300$, (c) $D = 500$ and (d) $D = 1{,}000$.

algorithm is superior in nine cases ($F4$, $F5$, $F6$, $F7$, $F8$, $F11$, $F13$, $F16$ and $F20$) in the search space with 1,000 dimensions.

One can observe with this results, that our proposal achieved better results than PSO and ABC algorithms, but as the number of dimensions increases (*e.g.* from 500 dimensions), the gain of performance of the algorithm is mitigated when compared to the APSO algorithm.

5.3 Initial Analysis of the ClanABeePSO Algorithm

This initial analysis shows the comparison of performance of ClanABeePSO with the ClanAPSO and ClanPSO algorithms. We used 100 dimensions and 1,000 iterations. This analysis was realized in five functions, the unimodal functions $F4$, $F7$ and $F19$, and the multimodal functions $F2$ and $F16$. We selected these functions in order to represent all classes of problems described in the entire benchmark.

In the conference of leaders, we executed the PSO algorithm and we tested two communication topologies, local and global. We adopted the following nomenclature for the ClanABeePSO based algorithms: "<Name of algorithm> - L" means the local communication topology and "<Name of algorithm>- G", global topology.

Table 5.10 presents the average fitness and (standard deviation) for local topology. One can observe that, in the most of cases, our proposal achieved better results than other techniques (the exception is $F2$ function). According to Table 5.11, we can conclude the same for the global topology. One can observe that the local topology achieved the best results in most of cases when compared to the global topology.

We also performed an analysis executing the APSO algorithm in the conference of leaders. We tested in all benchmark functions. Table 5.12 presents the results with the use the PSO algorithm in conference of leaders. The results for the PSO algorithm used in the conference of leaders are better than when the APSO algorithm is used (observe table 5.13). For the unimodal functions, the ClanABeePSO achieved better results than the ClanAPSO. For the multimodal functions, the ClanABeePSO algorithm did not achieved the desired performance for the Rastrigin-based functions ($F2$, $F5$, $F10$ and $F15$).

Table 5.2: Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 100 dimensions.

F#	ABeePSO	ABC	APSO	PSO
F1	**2.86E+08**	9.20E+09	3.23E+09	9.16E+08
	(8.22E+07)	(1.92E+09)	(1.19E+09)	(1.62E+08)
F2	9.58E+02	1.31E+03	**9.10E+02**	1.21E+03
	(63.09667)	(85.5529)	**(70.90304)**	(62.80075)
F3	**12.07975**	20.62107	19.62592	17.45967
	(0.656529)	(0.08368)	(0.401446)	(0.635132)
F4	**1.32E+14**	1.58E+15	3.11E+14	2.84E+14
	(3.05E+13)	(3.05E+14)	(1.24E+14)	(5.87E+13)
F5	**3.81E+08**	7.28E+08	4.45E+08	4.81E+08
	(2.64E+07)	(3.32E+07)	(4.70E+07)	(2.75E+07)
F6	**7.59E+06**	2.09E+07	1.97E+07	1.28E+07
	(8.71E+05)	(6.01E+04)	(3.27E+05)	(6.37E+05)
F7	**4.33E+10**	2.19E+11	8.45E+10	7.50E+10
	(9.10E+09)	(2.79E+10)	(1.61E+10)	(1.2E+10)
F8	**6.78E+13**	7.98E+16	2.99E+16	4.49E+14
	(5.49E+05)	(2.01E+16)	(9.34E+15)	(1.19E+14)
F9	**4.02E+08**	2.16E+09	1.15E+09	1.11E+09
	(9.68E+07)	(6.70E+08)	(4.64E+08)	(1.52E+08)
F10	9.73E+02	1.64E+03	**8.54E+02**	1.23E+03
	(39.83971)	(68.497)	**(77.301)**	(64.016)
F11	**25.93234**	41.67951	38.83987	34.68351
	(2.34576)	(0.132561)	(0.875927)	(0.87051)
F12	**9.26E+04**	5.86E+05	3.08E+05	1.95E+05
	(1.67E+04)	(7.60E+04)	(5.03E+04)	(2.19E+04)
F13	**8.08E+07**	6.46E+10	2.00E+10	4.69E+08
	(7.19E+07)	(1.52E+10)	(5.99E+09)	(1.19E+08)
F14	**6.29E+08**	5.36E+09	1.26E+09	1.32E+09
	(1.21E+08)	(9.68E+08)	(3.63E+08)	(2.21E+08)
F15	**9.82E+02**	1.75E+03	1.15E+03	1.24E+03
	(40.05412)	(57.49683)	(62.5245)	(67.00242)
F16	**25.48114**	42.02208	40.38697	35.10031
	(2.66121)	(0.098697)	(0.411129)	(0.811823)
F17	**1.84E+05**	6.79E+05	3.04E+05	2.77E+05
	(2.29E+04)	(7.42E+04)	(3.85E+04)	(3.40E+04)
F18	**1.02E+09**	2.01E+11	8.16E+10	1.16E+10
	(6.82E+08)	(3.15E+10)	(1.37E+10)	(3.25E+09)
F19	**2.09E+05**	8.01E+05	4.03E+05	3,08E+05
	(2.82E+04)	(7.82E+04)	(9.40E+04)	(3.96E+04)
F20	**1.09E+09**	2.58E+11	1.12E+11	1.17E+10
	(6.41E+08)	(4.56E+10)	(1.81E+10)	(5.35E+09)

Table 5.3: Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 300 dimensions.

F#	ABeePSO	ABC	APSO	PSO
F1	**1.41E+09**	6.12E+10	1.41E+09	6.14E+09
	(2.11E+08)	(4.48E+09)	(1.2E+09)	(7.23E+08)
F2	4.03E+03	5.40E+03	**1.53E+03**	4.39E+03
	(1.52E+02)	(1.43E+02)	**(1.27E+02)**	(1.26E+02)
F3	**13.81304**	21.04023	17.86329	20.49093
	(0.40062)	(0.041704)	(0.629742)	(0.160888)
F4	**5.88E+13**	2.44E+15	9.07E+13	1.62E+14
	(1.64E+13)	(5.44E+14)	(4.39E+13)	(3.97E+13)
F5	**3.29E+08**	8.14E+08	4.08E+08	4.11E+08
	(3.68E+07)	(3.59E+07)	(4.82E+07)	(2.45E+07)
F6	**4.81E+06**	2.08E+07	1.91E+07	9.62E+06
	(3.94E+05)	(9.46E+04)	(3.26E+05)	(5.56E+05)
F7	**2.03E+10**	2.01E+11	5.16E+10	4.51E+10
	(4.10E+09)	(2.92E+10)	(1.19E+10)	(6.51E+09)
F8	**3.13E+13**	6.78E+16	1.79E+16	6.3E+13
	(9.01E+12)	(1.56E+16)	(8.31E+15)	(1.75E+13)
F9	**2.42E+09**	6.09E+10	3.18E+09	6.66E+09
	(3.48E+08)	(5.72E+09)	(2.02E+09)	(7.21E+08)
F10	**8.50E+02**	6.06E+03	2.65E+03	4.59E+03
	(3.96E+01)	(1.27E+02)	(1.30E+02)	(1.12E+02)
F11	**60.82139**	84.81496	79.68142	74.01696
	(6.383927)	(0.08274)	(1.100449)	(1.241357)
F12	**6.02E+05**	2.31E+06	7.39E+05	1.06E+06
	(4.74E+04)	(1.70E+05)	(9.22E+04)	(5.85E+04)
F13	**4.14E+08**	6.71E+11	1.07E+11	9.34E+09
	(9.32E+07)	(5.86E+10)	(3.14E+10)	(2.64E+09)
F14	**2.39E+09**	3.96E+10	2.50E+09	7.28E+09
	(2.57E+08)	(5.34E+09)	(3.30E+08)	(7.06E+08)
F15	4.01E+03	6.00E+03	**3.16E+03**	4.49E+03
	(1.42E+02)	(1.18E+02)	**(1.47E+02)**	(1.04E+02)
F16	**96.59136**	127.2318	123.3011	118.5161
	(7.406842)	(0.128599)	(0.418354)	(1.745353)
F17	**8.41E+05**	4.76E+06	9.26E+05	1.45E+06
	(8.24E+04)	(6.23E+05)	(1.04E+04)	(1.59E+05)
F18	**4.23E+09**	1.35E+12	7.53E+10	4.70E+12
	(1.39E+09)	(1.01E+11)	(5.31E+10)	(8.02E+10)
F19	1.37E+06	6.82E+06	**1.30E+06**	2.11E+06
	(1.35E+05)	(7.44E+05)	**(3.48E+05)**	(2.41E+05)
F20	**6.04E+09**	1.51E+12	1.57E+11	1.78E+11
	(2.87E+09)	(1.18E+11)	(3.14E+10)	(2.19E+10)

Table 5.4: Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 500 dimensions.

F#	ABeePSO	ABC	APSO	PSO
F1	**2.17E+09**	1.22E+11	2.86E+09	1.19E+10
	(3.24E+08)	(7.33E+09)	(2.27E+09)	(8.12E+08)
F2	6.57E+03	9.91E+03	**2.86E+03**	7.59E+03
	(1.70E+02)	(2.04E+02)	**(2.27E+02)**	(1.27E+02)
F3	**14.58456**	21.17949	18.05753	20.77854
	(0.524085)	(0.043387)	(0.4994)	(0.058559)
F4	**4.81E+13**	2.30E+15	6.83E+13	1.28E+14
	(1.26E+13)	(4.41E+14)	(2.67E+13)	(2.84E+13)
F5	**3.09E+08**	7.68E+08	3.42E+08	3.97E+08
	(3.86E+07)	(3.67E+07)	(4.52E+07)	(2.40E+07)
F6	**4.47E+06**	2.09E+07	1.94E+07	8.73E+06
	(5.79E+05)	(9.32E+04)	(2.65E+05)	(3.13E+05)
F7	**1.41E+10**	2.36E+11	3.91E+10	3.58E+10
	(3.68E+09)	(3.26E+10)	(9.61E+09)	(5.45E+09)
F8	**9.43E+12**	1.22E+17	5.51E+15	2.64E+13
	(2.90E+12)	(1.88E+16)	(3.65E+15)	(8.72E+12)
F9	3.82E+09	1.25E+11	**2.67E+09**	1.52E+10
	(5.49E+08)	(9.68E+09)	**(5.02E+08)**	(1.20E+09)
F10	7.20E+03	1.06E+04	**4.21E+03**	7.87E+03
	(1.90E+02)	(1.39E+02)	**(1.48E+02)**	(1.34E+02)
F11	**99.34556**	127.5809	120.7821	115.0733
	(8.08220)	(0.10924)	(0.91662)	(1.05427)
F12	1.13E+06	5.69E+06	**8.53E+05**	2.05E+06
	(6.99E+04)	(4.68E+05)	**(8.87E+04)**	(1.11E+05)
F13	**5.21E+08**	1.30E+12	1.67E+10	4.00E+10
	(1.25E+08)	(7.93E+10)	(1.10E+10)	(9.53E+09)
F14	4.37E+09	1.04E+11	**3.89E+09**	1.55E+10
	(4.61E+08)	(7.25E+09)	**(2.89E+08)**	(1.00E+09)
F15	7.34E+03	1.07E+04	**6.43E+03**	8.01E+03
	(1.50E+02)	(1.13E+02)	**(2.41E+02)**	(1.20E+02)
F16	**166.3537**	212.7954	206.1731	205.4262
	(9.092158)	(0.119684)	(0.543216)	(1.845581)
F17	**1.46E+06**	1.05E+07	1.73E+06	2.91E+06
	(1.16E+05)	(1.30E+06)	(1.16E+05)	(2.04E+05)
F18	**1.38E+10**	2.77E+12	1.32E+11	3.48E+11
	(2.01E+09)	(1.59E+11)	(5.89E+10)	(2.90E+10)
F19	4.23E+06	1.75E+07	**3.85E+06**	5.40E+06
	(3.37E+05)	(1.98E+06)	**(8.90E+05)**	(4.68E+05)
F20	**1.43E+10**	3.03E+12	1.33E+11	4.24E+11
	(2.20E+09)	(1.39E+11)	(6.61E+10)	(3.70E+10)

Table 5.5: Performance comparison in terms of average fitness value and (standard deviation) between ABeePSO, PSO, APSO and ABC for all the 20 benchmark functions for 1,000 dimensions.

F#	ABeePSO	ABC	APSO	PSO
F1	**9.49E+09**	2.93E+11	1.02E+10	3.34E+10
	(7.41E+08)	(9.25E+09)	(4.10E+09)	(2.09E+09)
F2	14295.42	21862.81	**5646.419**	15878.37
	(305.095)	(414.0550)	**(251.3274)**	(201.1898)
F3	20.5246	21.3259	**18.2617**	20.9554
	(0.2398)	(0.0189)	**(0.3523)**	(0.0209)
F4	**5.55E+13**	2.29E+15	6.37E+13	1.05E+14
	(1.52E+13)	(4.13E+14)	(2.35E+13)	(2.39E+13)
F5	**2.73E+08**	8.38E+08	3.42E+08	3.38E+08
	(4.07E+07)	(2.99E+07)	(4.52E+07)	(2.58E+07)
F6	**1.36E+06**	2.08E+07	1.90E+07	6.41E+06
	(3.81E+05)	(9.16E+04)	(3.35E+05)	(4.34E+05)
F7	**1.22E+10**	6.17E+11	2.51E+10	2.38E+10
	(3.52E+09)	(2.13E+11)	(6.68E+09)	(5.68E+09)
F8	**3.34E+12**	9.58E+16	1.99E+15	6.62E+12
	(1.14E+12)	(1.49E+16)	(2.13E+15)	(2.83E+12)
F9	9.98E+09	3.07E+11	**4.47E+09**	3.89E+10
	(7.77E+08)	(1.43E+10)	**(9.55E+08)**	(2.91E+09)
F10	15171.77	22536.49	**8758.037**	16291.16
	(225.8321)	(250.1126)	**(234.3429)**	(223.1056)
F11	**193.5644**	234.4677	222.5258	219.8308
	(12.4481)	(0.1527)	(0.7495)	(2.6664)
F12	3.80E+06	1.86E+07	**1.71E+06**	4.76E+06
	(2.05E+05)	(1.71E+06)	**(1.55E+05)**	(2.12E+05)
F13	**3.61E+09**	2.87E+12	2.87E+10	1.75E+11
	(4.71E+08)	(1.03E+11)	(1.24E+10)	(1.88E+10)
F14	1.22E+10	3.03E+11	**7.41E+09**	4.09E+10
	(3.52E+09)	(1.95E+10)	**(5.23E+08)**	(2.52E+09)
F15	15229.42	22634.39	**12939.51**	16404.02
	(210.007)	(172.2981)	**(253.0937)**	(211.2478)
F16	**372.8392**	426.9702	413.1886	417.6073
	(9.7793)	(0.2063)	(0.9254)	(0.7588)
F17	4.07E+06	4.29E+07	**3.38E+06**	7.61E+06
	(3.43E+05)	(3.97E+06)	**(3.07E+05)**	(4.14E+05)
F18	2.96E+10	6.49E+12	**2.94E+10**	9.16E+11
	(4.17E+09)	(2.09E+11)	**(1.33E+10)**	(6.02E+10)
F19	1.42E+07	8.03E+07	**6.40E+06**	1.84E+07
	(1.16E+06)	(8.87E+06)	**(5.90E+05)**	(1.57E+06)
F20	**2.94E+10**	7.12E+12	3.88E+10	1.10E+12
	(4.18E+09)	(1.99E+11)	(1.24E+10)	(4.28E+10)

Table 5.6: Results of Wilcoxon test for the comparison of our proposal to the other approaches for $100D$ and $1,000$ iterations.

F#	ABC	APSO	PSO
F1	▲	▲	▲
F2	▲	▽	▲
F3	▲	▲	▲
F4	▲	▲	▲
F5	▲	▲	▲
F6	▲	▲	▲
F7	▲	▲	▲
F8	▲	▲	▲
F9	▲	▲	▲
F10	▲	▽	▲
F11	▲	▲	▲
F12	▲	▲	▲
F13	▲	▲	▲
F14	▲	▲	▲
F15	▲	▲	▲
F16	▲	▲	▲
F17	▲	▲	▲
F18	▲	▲	▲
F19	▲	▲	▲
F20	▲	▲	▲

Table 5.7: Results of Wilcoxon test for the comparison of our proposal to the other approaches for $300D$ and $3,000$ iterations.

F#	ABC	APSO	PSO
F1	▲	-	▲
F2	▲	▽	▲
F3	▲	▲	▲
F4	▲	▲	▲
F5	▲	▲	▲
F6	▲	▲	▲
F7	▲	▲	▲
F8	▲	▲	▲
F9	▲	-	▲
F10	▲	▲	▲
F11	▲	▲	▲
F12	▲	▲	▲
F13	▲	▲	▲
F14	▲	▲	▲
F15	▲	▽	▲
F16	▲	▲	▲
F17	▲	▲	▲
F18	▲	▲	▲
F19	▲	▽	▲
F20	▲	▲	▲

Table 5.8: Results of Wilcoxon test for the comparison of our proposal to the other approaches for $500D$ and $5,000$ iterations.

F#	ABC	APSO	PSO
F1	▲	-	▲
F2	▲	▽	▲
F3	▲	▲	▲
F4	▲	▲	▲
F5	▲	▲	▲
F6	▲	▲	▲
F7	▲	▲	▲
F8	▲	▲	▲
F9	▲	▽	▲
F10	▲	▽	▲
F11	▲	▲	▲
F12	▲	▽	▲
F13	▲	▲	▲
F14	▲	▽	▲
F15	▲	▽	▲
F16	▲	▲	▲
F17	▲	▲	▲
F18	▲	▲	▲
F19	▲	▽	▲
F20	▲	▲	▲

Table 5.9: Results of Wilcoxon test for the comparison of our proposal to the other approaches for $1000D$ and $10,000$ iterations.

F#	ABC	APSO	PSO
F1	▲	-	▲
F2	▲	▽	▲
F3	▲	▽	▲
F4	▲	▲	▲
F5	▲	▲	▲
F6	▲	▲	▲
F7	▲	▲	▲
F8	▲	▲	▲
F9	▲	▽	▲
F10	▲	▽	▲
F11	▲	▲	▲
F12	▲	▽	▲
F13	▲	▲	▲
F14	▲	▽	▲
F15	▲	▽	▲
F16	▲	▲	▲
F17	▲	▽	▲
F18	▲	-	▲
F19	▲	▽	▲
F20	▲	▲	▲

Table 5.10: Comparison of ClanABeePSO, ClanAPSO and ClanPSO algorithms with conference of leaders executing a Local PSO.

F#	ClanABeePSO - L	ClanAPSO - L	ClanPSO - L
$F2$	1031.6137 (56.0135)	**771.6618 (101.8197)**	1249.418 (61.9373)
$F4$	**1.56E+14 (4.1E+13)**	2.66E+14 (1.04E+04)	4.97E+14 (1.62E+14)
$F7$	**3.99E+10 (1.06E+10)**	7.34E+10 (1.44E+10)	8.87E+10 (1.86E+10)
$F16$	**30.1947 (1.5911)**	39.5529 (0.8987)	40.6395 (0.27026)
$F19$	**2.18E+05 (2.85E+04)**	3.03E+05 (6.91E+04)	3.56E+05 (6.01E+04)

Table 5.11: Comparison of ClanABeePSO, ClanAPSO and ClanPSO algorithms with conference of leaders executing a global PSO.

F#	ClanABeePSO - G	ClanAPSO - G	ClanPSO - G
$F2$	1026.3147 (58.1114)	**838.1519 (71.2261)**	1291.227 (64.2355)
$F4$	**1.65E+04 (3.54E+13)**	2.87E+14 (1.00E+14)	4.99E+14 (1.94E+14)
$F7$	**3.85E+10 (6.81E+09)**	8.58E+10 (1.82E+10)	9.61E+10 (2.11E+10)
$F16$	**30.3466 (1.5815)**	39.6915 (0.6011)	40.7952 (0.3673)
$F19$	**2.29E+05 (3.57E+04)**	3.32E+05 (7.83E+04)	3.39E+05 (8.67E+04)

Table 5.12: Performance comparison in terms of average fitness value and (standard deviation) between ClanABeePSO and ClaAPSO with execution of PSO in the conference of leaders for all the 20 benchmark functions for 100 dimensions.

F#	ClanABeePSO - L	ClanABeePSO - G	ClanAPSO - L	ClanAPSO - G
F1	5.45E+08 (1.08E+08)	**5.29E+08 (1.33E+08)**	2.61E+09 (1.14E+09)	2.42E+09 (1.03E+09)
F2	1031.6137 (56.0135)	1026.3147 (58.1114)	**771.6618 (101.8197)**	838.1519 (71.2261)
F3	13.1624 (0.5596)	**12.5989 (0.4668)**	19.0352 (0.6809)	19.3434 (0.5242)
F4	**1.56E+14 (4.1E+13)**	1.65E+14 (3.54E+13)	2.66E+14 (1.04E+14)	2.87E+14 (1.00E+14)
F5	4.2E+08 (2.35E+07)	4.17E+08 (2.48E+07)	**3.50E+08 (4.72E+07)**	3.92E+08 (4.79E+07)
F6	1.04E+07 (8.58E+05)	**9.41E+06 (5.57E+05)**	1.87E+07 (9.67E+05)	1.89E+07 (8.04E+05)
F7	3.99E+10 (1.06E+10)	**3.85E+10 (6.81E+09)**	7.34E+10 (1.44E+10)	8.58E+10 (1.82E+10)
F8	1.14E+14 (7.28E+13)	**7.22E+13 (2.33E+13)**	2.09E+16 (9.78E+15)	2.32E+16 (1.05E+16)
F9	7.62E+08 (2.51E+08)	**5.8E+08 (1.07E+08)**	1.11E+09 (5.34E+08)	1.64E+09 (5.92E+08)
F10	1050.3797 (52.3948)	1021.8675 (51.4933)	**704.9859 (87.0795)**	930.2839 (83.4512)
F11	29.8896 (1.8033)	**29.5806 (1.6640)**	37.6465 (1.3326)	38.5982 (0.8247)
F12	**7.77E+04 (9.97E+03)**	1.04E+05 (2.09E+04)	2.59E+05 (4.19E+04)	2.73E+05 (4.84E+04)
F13	**8.12E+07 (2.12E+07)**	1.29E+08 (1.09E+08)	1.61E+10 (5.28E+09)	1.58E+10 (6.89E+09)
F14	**8.09E+08 (1.3E+08)**	8.52E+08 (1.45E+08)	1.14E+09 (3.50E+08)	1.18E+09 (3.73E+08)
F15	1070.2954 (48.2117)	1067.4445 (53.4160)	**988.4879 (79.7370)**	1012.5407 (79.9971)
F16	**30.1947 (1.5911)**	30.3466 (1.5815)	39.5529 (0.8987)	39.6915 (0.6011)
F17	**1.91E+05 (2.50E+04)**	2.08E+05 (3.21E+04)	2.65E+05 (5.46E+04)	2.70E+05 (4.30E+04)
F18	**8.92E+08 (1.9E+08)**	1.25E+09 (5.44E+08)	7.83E+10 (1.65E+10)	7.61E+10 (1.46E+10)
F19	**2.18E+05 (2.85E+04)**	2.29E+05 (3.57E+04)	3.03E+05 (6.91E+04)	3.32E+05 (7.83E+04)
F20	**7.07E+08 (1.59E+08)**	1.19E+09 (6.34E+08)	1.04E+11 (2.04E+10)	1.00E+11 (2.19E+10)

Table 5.13: Performance comparison in terms of average fitness value and (standard deviation) between ClanABeePSO and ClaAPSO with execution of APSO in the conference of leaders for all the 20 benchmark functions for 100 dimensions.

F#	ClanABeePSO - L	ClanABeePSO - G	ClanAPSO - L	ClanAPSO - G
F1	**2.14E+09** **(8.18E+08)**	2.21E+09 (8.77E+08)	3.56E+09 (1.38E+09)	3.33E+09 (1.29E+09)
F2	**818.9345** **(62.1110)**	854.3629 (72.2428)	833.5397 (80.1530)	845.7962 (88.5562)
F3	19.5079 (0.3109)	19.5476 (0.2776)	**19.4254** **(0.3693)**	19.4973 (0.3233)
F4	**2.21E+14** **(9.49E+13)**	2.5E+14 (9.76E+13)	3.84E+14 (1.52E+14)	4.12E+14 (1.79E+14)
F5	**3.85E+08** **(5.14E+07)**	3.97E+08 (5.20E+07)	4.14E+08 (5.72E+07)	4.11E+08 (5.47E+07)
F6	**1.94E+07** **(3.89E+05)**	1.95E+07 (3.28E+05)	1.95E+07 (3.34E+05)	1.96E+07 (3.54E+05)
F7	**7.6E+10** **(1.63E+10)**	8.07E+10 (1.68E+10)	9.69E+10 (1.97E+10)	1.01E+11 (2.69E+10)
F8	2.16E+16 (9.48E+15)	**2.14E+16** **(8.29E+15)**	3.14E+16 (1.21E+16)	2.93E+16 (8.79E+15)
F9	1.58E+09 (5.22E+08)	**1.53E+09** **(4.9E+08)**	2.82E+09 (1.08E+09)	2.47E+09 (6.68E+08)
F10	914.0123 (105.0599)	936.1181 (95.7550)	**887.2911** **(113.8297)**	928.4704 (82.8355)
F11	38.8131 (0.8369)	38.7006 (0.7980)	**38.3081** **(1.0786)**	38.7691 (0.7347)
F12	**2.51E+05** **(3.72E+04)**	2.59E+05 (4.30E+04)	3.14E+05 (4.55E+04)	3.48E+05 (5.33E+04)
F13	1.38E+10 (5.15E+09)	**1.27E+10** **(4.36E+09)**	1.95E+10 (6.57E+09)	1.94E+10 (5.98E+09)
F14	**1.09E+09** **(3.73E+08)**	1.24E+09 (3.48E+08)	1.28E+09 (4.40E+08)	1.54E+09 (4.73E+08)
F15	1026.1148 (67.3962)	1043.2662 (90.3853)	**1019.0113** **(70.6286)**	1025.8771 (91.8249)
F16	39.8903 (0.3489)	**39.6015** **(0.3553)**	39.7034 (0.3414)	39.7101 (0.4693)
F17	2.72E+05 (4.79E+04)	**2.62E+05** **(4.61E+04)**	3.26E+05 (6.58E+04)	3.19E+05 (7.37E+04)
F18	7.49E+10 (1.36E+10)	**7.19E+10** **(1.37E+10)**	8.24E+10 (1.46E+10)	8.47E+10 (1.27E+10)
F19	**3.02E+05** **(6.55E+04)**	3.12E+05 (6.05E+04)	3.75E+05 (9.05E+04)	3.82E+05 (8.33E+04)
F20	1.08E+11 (2.14E+10)	**1.02E+11** **(1.82E+10)**	1.19E+11 (2.23E+10)	1.19E+11 (1.85E+10)

Chapter 6

Final Considerations and Future Works

This chapter presents the conclusions, the discussions and future works.

6.1 Conclusions and Discussions of the Results

The Adaptive Bee and Particle Swarm Optimization (ABeePSO) is a combined algorithm that adopts an operator based on artificial bees to generate diversity for Particle Swarm Optimization with adaptive behavior (APSO). The APSO algorithm was chosen because it has a good convergence capability, but as the number of dimensions increases the algorithm loses the ability to maintain the diversity. However, a new algorithm with suitable exploitation-exploration tradeoff was obtained when the operator based on artificial bees was included in the APSO algorithm.

The proposed diversity operator generates the diversity by allowing the particles to adopt the behavior of bees. A half of the swarm are guide bees in the exploration mode and the other ones are guided bees. The movement of the guide bees was modified by using the barycenter of the swarm, which was inspired in the Fish School Search algorithm, and the dispersion step. The guide bees are spread in the search space, while the Elitist Learning Strategy mechanism moves just one particle per iteration. Thus, the proposed operator can generate more diversity in a lower number of iterations than the strategy presented in the APSO algorithm, without losing the coherence with previous search processes.

The analysis of the parameters is necessary to realize the adaptations of the proposed algorithm. In the analysis of the acceleration coefficients, we observed that it is not necessary to change the original approach proposed for the APSO. The acceleration rate (δ) is the same. The analysis with some functions of intervals of fuzzy membership functions presented that it is necessary to decrease the *Convergence* interval and to increase the *Exploration* interval. This modification allows a reduction in the number of executions of the diversity operator.

The analysis of the ABeePSO algorithm parameters showed that these parameters values influence mainly for multimodal functions. The analysis of dispersion step was performed to adapt the behavior of the logistic function to the evolutionary factor values. The analysis of the stagnation counter is preliminary but it showed that the used value must be low. With low values, the algorithm presents a better exploitation ability.

In the analysis of the results, we can observe that the presence of ABC based operator allows a better exploration ability of the search space. In the analysis of convergence, our proposal achieved better results than the other classical approaches, with the exception of functions $F2$ and $F10$. In the analysis of dimensionality, the gain of performance obtained by the ABeePSO algorithm is mitigated for extremely high dimensionality (*e.g.* from 500 dimensions). The statistical tests confirm that we obtained better results than the original approaches, APSO and ABC.

We also tried to change the topology to include sub-swarm decentralized control by using the Clan concept. The initial results for the ClanABeePSO algorithm are very encouraging since it helps to maintain the particles spread over the search process. We observed that the APSO algorithm is not the best option for the conference of leaders. It is better to use the PSO algorithm.

6.2 Future Works

This research can be extended in several lines, for example:

- Realize a deeper analysis of the stagnation counter and the other parameters of the ClanABeePSO algorithm;
- Investigate the use of dynamic clans, in which it can use the adaptive behavior. One can try to use the characteristics of the guide bees to realize the migration of particles or eliminate a clan and generate other one in a new region of the search space;
- Propose other types of combinations with another approaches based on swarm (for example, fireflies and bacterias), aiming to generate more heterogeneous swarms;
- Apply these techniques in real world problems, such as: optimization of machine learning algorithms or supply chain management;
- Develop the proposed algorithms in parallel platform and realize a comparative study with CPUs and GPUs.

Appendix A

Publications:

Title: A Cooperative Approach using Particle Swarm Optimization with Adaptive Behavior.

Authors: L. N. Vitorino, S. F. Ribeiro and C. J. A. Bastos-Filho.

Published in: 10th Brazilian Congress on Computational Intelligence (CBIC'2011), pages 1-8, 2011.

Abstract: Particle Swarm Optimization (PSO) has been widely used to solve real world optimization problems. Since the original PSO fails to maintain the diversity along the search process, some PSO variations have been proposed to overcome this limitation. The recently introduced Adaptive Particle Swarm Optimization (APSO) presents faster convergence, while maintaining the population diversity, by including into the PSO algorithm the auto-adaptation of the parameters according to the current spatial distribution of the swarm. Besides, there are some PSO variations which incorporate a cooperative behavior, such as the Clan Particle Swarm Optimization (ClanPSO). In this paper, we propose to include the auto-adaptation ability solely in the Clans of the ClanPSO algorithm and compare its performance to previous approaches. We evaluated our proposal in six benchmark functions recently proposed in IEEE Congress on Evolutionary Computation 2010. Our proposal obtained similar or better results in terms of fitness when compared to the original ClanAPSO, but presents a lower computational cost. We obtained better results when compared to other cooperative approaches and the classical PSO approaches.

Title: A Hybrid Swarm Intelligence Optimizer based on Particles and Artificial Bees for High-Dimensional Search Spaces.

Authors: L. N. Vitorino, S. F. Ribeiro and C. J. A. Bastos-Filho.

Published in: IEEE Congress on Evolutionary Computation 2012 (CEC'2012), 381-386, 2012.

Abstract: Real-world problems can present search spaces with hundreds of dimensions and swarm intelligence algorithms have been developed to solve this type of problems. Particle Swarm Optimization (PSO) presents a fast convergence in continuous problems, but it cannot maintain diversity along the search process. On the other hand, Artificial Bee Colony (ABC) presents the capability to generate diversity when the guide bees are in the exploration mode. We propose in this paper to introduce a mechanism based on the ABC to generate diversity in an adaptive PSO approach and analyze its performance in high dimensional search spaces. The swarm switches its behavior depending on the dispersion of the swarm. We evaluated our proposal in a well known set of 20 benchmark functions recently proposed in 2010 and our proposal achieved better performance than PSO, APSO and ABC in most of the cases.

Title: A Mechanism based on Artificial Bee Colony to Generate Diversity in Particle Swarm Optimization.

Authors: L. N. Vitorino, S. F. Ribeiro and C. J. A. Bastos-Filho.

Published in: The 3rd International Congress on Swarm Intelligence, 2012.

Abstract: Particle Swarm Optimization (PSO) presents a fast convergence in continuous problems, but it cannot maintain diversity along the entire search process. On the other hand, Artificial Bee Colony (ABC) presents the capability to generate diversity when the guide bees are in the exploration mode. We propose in this paper to introduce a mechanism based on the ABC to generate diversity when the PSO stagnates. The swarm entities switches between two pre-defined behaviors depending on the evolutionary state. We used the Adaptive PSO variation since it presents better adaptation capabilities and it also has a mechanism to evaluate the evolutionary state. We evaluated our proposal for all benchmark functions recently proposed for large scale optimization in CEC2010 and our proposal achieved better performance in most of the cases.

Observation: This paper was selected to Special Issue entitled ”Swarm Intelligence for Neural Networks” (Special Issue: SINN 2012) in Neurocomputing (famous SCI-indexed Journal, IF=1.429).

Bibliography

[AGH05] K. Arnald, J. Gosling, and D. Holmes. *The Java Programming Language, Fourth Edition*. Addison Wesley Professional, 2005.

[AMZ09] R. Akbari, A. Mohammadi, and K. Ziarati. A powerful bee swarm optimization algorithm. *INMIC'09. IEEE 13th International Multitoptic Conference*, pages 1–6, 2009.

[AMZ10] R. Akbari, A. Mohammadi, and K. Ziarati. A novel bee swarm optimization algorithm for numerial function optimization. In *Comunications in Nonlinear Science and Numerial Simulation*, volume 15, pages 3142–3155. Elsevier, 2010.

[Bey01] H. Beyer. *The Theory of Evolution Strategies*. Springer, 2001.

[BFCFdM09] C. J. A. Bastos-Filho, D. F. Carvalho, E. M. N. Figueiredo, and P. B. C. de Miranda. Dynamic clan particle swarm optimization. *2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 249–254, 2009.

[BFCMC08] C. J. A. Bastos-Filho, M. P. Caraciolo, P. B. C. Miranda, and D. F. Carvalho. Multi-ring particle swarm optimization. *Proceedings of the 10th Brazilian Symposium on Neural Networks*, pages 111–116, 2008.

[BFLNL$^+$08] C. J. A. Bastos-Filho, F. B. Lima-Neto, A. J. C. C. Lins, A. I. S. Nascimento, and M. P. Lima. A novel search algorithm based on fish school behavior. *IEEE International Conference on Systems, Man, and Cybernetics*, pages 2646–2651, 2008.

[BFNS$^+$09] C. J. A. Bastos-Filho, F. B. Lima Neto, M. F. C. Sousa, M. R. Pontes, and S. S. Madeiro. On the influence of the swimming operators in the fish school search algorithm. *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 5012–5017, 2009.

[BK06] B. Basturk and D. Karaboga. An artificial bee colony (abc) algorithm for numeric function optimization. *IEEE Swarm Intelligence Symposium*, pages 12–14, 2006.

[BK07] D. Bratton and J. Kennedy. Defining a standard for particle swarm optimization. *IEEE Swarm Intelligence Symposium (SIS'2007)*, pages 120–127, 2007.

[BS02] H. Beyer and H. Schwefel. *Evolution Strategies - A comprehensive introduction.* Springer, 2002.

[BW89] G. Beni and J. Wang. Swarm intelligence in cellular robotic systems. *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, 102:703–704, 1989.

[Cam12] R. Campbell. Aquariumpros, inc. minnesota - aquarium maintenance service and sales. `http://aquariumprosmn.com/tag/saltwater/page/2/`, May 2012. Online; Accessed: 05/15/2012.

[CBF08] D. F. Carvalho and C. J. A. Bastos-Filho. Clan particle swarm optimization. *IEEE Congress on Evolutionary Computation, 2008*, pages 3044–3051, 2008.

[CK02] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[Cle99] M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC'99*, 3, 1999.

[Cor11] OriginLab Corporation. Originlab - origin and originpro - data analysis and graphing software. `http://www.originlab.com`, January 2011. Online; Accessed: 01/10/2011.

[DB05] M. Dorigo and C. Blum. Ant colony optimization theory: A survey. In *Theoretical Computer Science*, volume 344, pages 243–278. Elsevier, 2005.

[DBDA09] S. Das, A. Biswas, S. Dasgupta, and A. Abraham. Bacterial forafing optimization algorithm: Theoretical foundations, analysis and applications. *Studies in Computational Intelligence*, pages 23–55, 2009.

[DC99] M. Dorigo and G. Di Caro. Ant colony optimization: A new metaheuristic. In *Proceedings of the Congress on Evolutionary Computation*, pages 1470–1477. IEEE Press, 1999.

[DJW02] S. Droste, T. Jansen, and I. Wegener. Optimization with randomized search heuristics - the (a)nfl theorem, realistic scenarios and difficult functions. *Theoretical Computer Science*, 287:131–144, 2002.

[Dor97] M. Dorigo. Ant colony system: a cooperative learning approach to the traveling salesman problem. In *IEEE Transactions on Evolutionary Computation*, pages 53–66. IEEE, 1997.

[dSC10] [In Portuguese] R. M. C. de S. Castro. Uma ferramenta computacional para analise de algoritmos de otimizacao para sistemas dinamicos baseados em inteligencia de enxames. Master's thesis, Escola Politecnica de Pernambuco - Universidade de Pernambuco (POLI - UPE), 2010.

[Eco12] The Ecologist. Heavy bee colony losses in us could lead to price rise. `http://www.theecologist.org/News/news_round_up/477710/heavy_bee_colony_losses_in_us_could_lead_to_price_rise.html`, May 2012. Online; Accessed: 05/15/2012.

[EK95a] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, 1995.

[EK95b] R. Eberhart and J. Kennedy. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Service Center, 1995.

[Fou11a] The Apache Software Foundation. Maven - welcome to apache maven. `http://maven.apache.org`, January 2011. Online; Accessed: 01/10/2011.

[Fou11b] The Eclipse Foundation. Eclipse - the eclipse foundation open source community website. `http://www.eclipse.org`, January 2011. Online; Accessed: 01/10/2011.

[GWH10] P. Guo, X. Wang, and Y. Han. The enhanced genetic algorithms for the optimization design. *3rd International Conference on Biomedical Engineering and Informatics (BMEI)*, 7:2990–2994, 2010.

[GyMg10] L. Gao-yang and L. Ming-guang. The summary of differential evolution algorithm and its improvements. *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, 3:153–156, 2010.

[HKK$^+$10] M. E. Houle, H. Kriegel, P. Kroger, E. Schubert, and A. Zimek. Can shared-neighbor distances defeat the curse of dimensionality? *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management*, pages 482–500, 2010.

[KA07] D. Karaboga and B. Akay. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39:459–471, 2007.

[KA09a] D. Karaboga and B. Akay. A compartive study of artificial bee colony algorithm. *Applied Mathematics and Computation - AMC*, 214:108–132, 2009.

[KA09b] D. Karaboga and B. Akay. *A survey: algorithms simulating bee swarm intelligence*, volume 31, pages 61–85. Springer, 2009.

[Kar05] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[KES01] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, 340 Pine Street, Sixth Floor, San Francisco, CA, USA, 2001.

[KG09] K. N. Krishnanand and Debasish Ghose. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence*, pages 87–124, 2009.

[KK12] O. Kovarik and P. Kordik. Max-min ant system with linear memory complexity. In *Proceedings of the 2012 Congress on Evolutionary Computation, CEC'12*, pages 1469–1473. IEEE, 2012.

[KM02] J. Kennedy and R. Mendes. Population structure and particle swarm performance. *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02*, pages 1671–1676, 2002.

[KM06] J. Kennedy and R. Mendes. Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, pages 515–519, 2006.

[LBFN$^+$12] A. J. C. C. Lins, C. J. A. Bastos-Filho, D. N. O. Nascimento, M. A. C. Oliveira Junior, and F. B. Lima Neto. *Analysis of Performance of the Fish School Search Algorithm Running in Graphic Processing Units*, pages 17–32. 2012.

[LFLW02] W. Z. Lu, H. Y. Fan, A. Y. T. Leung, and J. C. K. Wong. Analysis of pollutant levels in central hong kong applying neural network method with particle swarm optimization. *Environmental Monitoring and Assessment*, 79(3):217–230, 2002.

[LJG06] Z. Lian, B. Jiao, and X. Gu. A similar particel swarm optimization algorithm for jo-shop scheduling to minimize makespan. *Applied Mathematics and Computation*, 183:1008–1017, 2006.

[Low11] R. Lowry. Concepts & applications of inferential statistics. `http://vassarstats.net/textbook/`, November 2011. Online; Accessed: 11/16/2011.

[Mit98] M. Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, 1998.

[MM12] R. Murugan and M. R. Mohan. Artificial bee colony optimization for the combined heat and power economic dispatch problem. *ARPN Journal of Engineering and Applied Sciences*, 7(5):597–604, 2012.

[MMAK02] S. D. Muller, J. Marchetto, S. Airaghi, and P. Kournoutsakos. Optimization based on bacterial chemotaxis. *IEEE Transactions on Evolutionary Computation*, 6:16–29, 2002.

[PGK$^+$05] D. T. Pham, A. Ghanbarzadeh, E. Ko, S. Otri, S. Rahim, and M. Zaidi. The bees algorithm. Technical report, Manufacturing Engineering Center, Cardiff University, Cardiff CF24 3AA, UK, 2005.

[PLNBF11] M. R. Pontes, F. B. Lima-Neto, and C. J. A. Bastos-Filho. Adaptive clan particle swarm optimization. *Proceedings of IEEE Swarm Intelligence Symposium*, pages 17–22, 2011.

[POA$^+$07] D. T. Pham, S. Otri, A. Afify, M. Mahmuddin, and H. Al-Jabbouli. Data clustering using the bees algorithm. *40th CIRP International Manufacturing Systems Seminar, Liverpool*, 2007.

[Pow07] W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* John Wiley & Sons, Hoboken, New Jersey, 2007.

[PSL05] K. Prince, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Pratical Approach to Global Optimization.* Springer, 2005.

[RM87] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* The MIT Press, 1987.

[RNI10] M. Radovanovic, A. Nanopoulos, and M. Ivanovic. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research 11*, pages 2487–2531, 2010.

[SE98] Y. Shi and R. Eberhart. A modified particle swarm optimizer. *Proceedings of 1998 IEEE International Conference Evolutionary Computation*, pages 69–73, 1998.

[Ser09] [In Portuguese] A. B. S. Serapiao. Fundamentos de otimizacao por inteligencia de enxames: uma visao geral. *SBA: Controle & Automacao Sociedade Brasileira de Automatica*, 20:271–304, 2009.

[Sig12] Y. Sigurdardottir. Murder is everywhere - seven renowned crime writers blog from different corners of the world. `http://http://murderiseverywhere.blogspot.com.br/2011/01/coincidence.html`, May 2012. Online; Accessed: 05/15/2012.

[SLS12] T. Schoene, S. A. Ludwig, and R. J. Spiteri. Step-optimized particle swarm optimization. *Proceedings of the 2012 Congress on Evolutionary Computation, CEC'12*, pages 654–662, 2012.

[SNHD^{+}05] P. N. Suganthan, J. J. Liang N. Hansen, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University (NTU), Singapore, 2005.

[TD05] D. Teodorovic and M. Dell'orco. Bee colony optimization: A cooperative learning approch to complex transportation problems. In *Proceedings of the 16th Mini-EURO Conference on Advanced OR and AI Methods in Transportation*, pages 51–60, 2005.

[Teo09] D. Teodorovic. Bee colony optimization: Principles and applications. *8th Seminar on Neural Network Applications in Eletrical Engineering*, pages 151–156, 2009.

[TLS^{+}10] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Benchmark functions of the cec'2010 special session and competition on large scale global optimization. Technical report, University of Science and Technology of China (USTC), School of Computer Science and Technology, Nature Inspired Computation and Applications Laboratory (NICAL): China, 2010.

[VRBF11] L. N. Vitorino, S. F. Ribeiro, and C. J. A. Bastos-Filho. A cooperative approach using particle swarm optimization with adaptive behavior. *10th Brazilian Congress on Computational Intelligence (CBIC'2011)*, pages 1–8, 2011.

[WM95] D. H. Wolpert and W. G. Macready. No free lunch theorems for search. Technical report, Santa Fe Institute, 1995.

[WM97] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.

[WSZ^{+}04] M. P. Wachowiak, R. Smolíková, Y. Zheng, J. M. Zurada, and A. S. Elmaghraby. An approach to multimodal biomedical image registration utilizing particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):289–301, 2004.

[Yan09] X.-S. Yang. Forefly algorithms for multimodal optimization. In *Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications*, pages 169–178. Springer, 2009.

[YATNM12] D. Yazdani, M. R. Akbarzadeh-Totonchi, B. Nasiri, and M. R. Meybodi. A new artificial fish swarm algorithm for dynamic optimiazation. *Proceedings of the 2012 Congress on Evolutionary Computation, CEC'12*, pages 472–479, 2012.

[ZAZ11] K. Ziarati, R. Akbari, and V. Zeighami. On the performance of bee algorithms for resource-constrained project scheduling problem. *Applied Soft Computing*, 11:3720–3733, 2011.

[Zim01] H.-J. Zimmermann. *Fuzzy Set Theory and its Applications*. Springer, 2001.

[ZZLC09] Z. Zhan, J. Zhang, Y. Li, and H. Chung. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, 39(6):1362–1381, 2009.