

SEP**TNM****INSTITUTO TECNOLÓGICO DE TIJUANA****DIVISIÓN DE ESTUDIOS DE POSGRADO E
INVESTIGACIÓN****MODELO MULTI-METAHEURÍSTICO COMPETITIVO PARA
LA OPTIMIZACIÓN DE SISTEMAS DIFUSOS****TRABAJO DE TESIS****Presentado por
MARYLU LARA LAGUNES****Para Obtener el Grado de
DOCTOR EN CIENCIAS EN COMPUTACIÓN****Director de Tesis
DR. OSCAR CASTILLO LÓPEZ****Co-Director de Tesis
DR. JOSÉ LUCIANO SORIA ARTECHE
DR. FEVRIER VALDEZ ACOSTA****Tijuana, BC, Octubre, 2021**



**INSTITUTO TECNOLÓGICO DE TIJUANA
POSGRADO EN COMPUTACION**

Asunto: Se autoriza impresión de Trabajo de Tesis

Tijuana, B.C., 19 de Octubre del 2021

C. M.C. MARIBEL GUERRERO LUIS

Jefe del Depto. de Servicios Escolares
Presente.

En lo referente al trabajo de tesis escrito, con título "**MODELO MULTI-METAHEURÍSTICO COMPETITIVO PARA LA OPTIMIZACIÓN DE SISTEMAS DIFUSOS**", presentado por la **C. MARYLU LARA LAGUNES**, alumna del Doctorado en Ciencias en Computación con número de control **D09210201**, informamos a usted que después de una minuciosa revisión, de acuerdo con lo establecido en el reglamento vigente para este caso, nuestro dictamen es: Se aprueba en todas sus partes, en virtud de reunir los requisitos de un trabajo de grado de Doctorado y a la vez se autoriza al interesado para que proceda de inmediato a la impresión del mismo.

ATENTAMENTE

DR. OSCAR CASTILLO LOPEZ
PRESIDENTE

DR. JOSE LUCIANO SORIA ARTECHE
SECRETARIO

DR. FEVRIER ADOLFO VALDEZ ACOSTA
VOCAL

DR. JOSE MARIO GARCIA VALDEZ
VOCAL

DR. PROMETEO CORTES ANTONIO
VOCAL

c.c.p. Oficina de Titulación
c.c.p. División de Estudios de Posgrado e Investigación
c.c.p. Expediente
c.c.p. Interesado

EPMO/*inf





Instituto Tecnológico de Tijuana

Tijuana, Baja California,
22/octubre/2021

OFICIO No. 105/DEPI/2021

Asunto: **Autorización de Impresión de Tesis**

MARIBEL GUERRERO LUIS
JEFA DEL DEPARTAMENTO DE SERVICIOS ESCOLARES
PRESENTE

En lo referente al trabajo de tesis, "Modelo multi-metaheurístico competitivo para la optimización de sistemas difusos" Presentado por C. **Marylu Lara Lagunés**, alumna del Doctorado en Ciencias en Computación con número de control **D09210201**; informo a usted que a solicitud del comité de tutorial, tengo a bien **Autorizar la impresión de Tesis**, atendiendo las disposiciones de los Lineamientos para la Operación de Estudios de Posgrado del Tecnológico Nacional de México.

Sin más por el momento le envío un cordial saludo.

A T E N T A M E N T E

*Excelencia en Educación Tecnológica.
Por una juventud integrada al desarrollo de México.*

GUADALUPE HERNANDEZ ESCOBEDO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACION



**DIVISIÓN DE ESTUDIOS DE POSGRADO
E INVESTIGACIÓN**

ccp. Archivo
GHE/lap



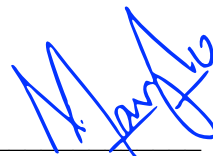
CARTA DECLARACIÓN DE PROPIEDAD INTELECTUAL

Tijuana, BC a 20 de octubre del 2021

Yo MARYU LARA LAGUNES reconozco que el Trabajo de Tesis de Doctorado que realice durante mis estudios en el Doctorado en Ciencias en Computación del Instituto Tecnológico de Tijuana fue parte del Proyecto de Investigación Titulado: **Modelo Multi-Metaheurístico Competitivo para la Optimización de Sistemas Difusos** NÚMERO 0122 que desarrolla mi director de tesis el Dr. Oscar Castillo López y del cual es responsable del proyecto de investigación. Por esta razón, los métodos, modelos, algoritmos, y software realizados, así como datos y resultados obtenidos durante el desarrollo de mi tesis de doctorado son propiedad intelectual de mi Director y Codirector de Tesis, del Tecnológico Nacional de México, Instituto Tecnológico de Tijuana y del Conacyt, y No podré utilizarlos por mi cuenta durante, Ni después de terminar mi beca o estudios, excepto a solicitud escrita para poder utilizarlos bajo una colaboración directa con mi director el cual es responsable del proyecto de investigación. Por tanto, estoy de acuerdo en que No podre utilizar ni tomar modelos, ni datos utilizados en este proyecto de investigación y en el desarrollo de tesis para: presentaciones, publicaciones ni desarrollo de mi propia investigación que pudiera desarrollar una vez concluidos mis estudios.

Atentamente

Marylu Lara Lagunes



Estudiante de Doctorado en Ciencias en Computación

DECLARACIÓN DE ORIGINALIDAD

Tijuana, BC., 21 de Octubre de 2021,

Yo, **Marylu Lara Lagunes**, estudiante del Doctorado en Ciencias en Computación, en mi calidad de autor manifiesto que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patente y similaridad. Por lo tanto, la obra realizada es de mi exclusiva autoría y no infringí en copiar el texto o imágenes, de fuentes de información por lo cual soy responsable del escrito que aquí se presenta.

Así mismo, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

En caso de presentarse cualquier reclamación o acción por parte de terceros en cuanto a los derechos de autor sobre la obra en cuestión, acepto toda la responsabilidad de tal infracción y relevo de esta a mi director de tesis, así como al Tecnológico Nacional de México, al Instituto Tecnológico de Tijuana y a sus respectivas autoridades.



Marylu Lara Lagunes
Estudiante de Doctorado en Ciencias en Computación

Resumen

La lógica difusa se basa en la toma de decisiones con diferentes grados de verdad, donde es aplicable a cualquier sistema, sin importar si existe o no un modelo matemático, ya que es muy flexible en cuanto a razonamiento basado en el pensamiento humano. Esta siendo utilizada en electrodomésticos básicos como una lavadora, hasta tecnología de punta como los carros tesla.

En la computación suave también existen combinaciones para realizar hibridaciones potentes, que ayuden a generar un mejor rendimiento en sus tareas, como lo es el uso de algoritmos bio-inspirados para la optimización de controladores difusos, por ello el presente trabajo de tesis propone un modelo competitivo multi-metaheurístico para la optimización de sistemas difusos, el cual determine a base de experimentación que método es el que tiene mayor eficacia y rendimiento de un controlador.

En este caso de estudio se utilizan una serie de algoritmos, que competitivamente cada uno de ellos evalúan al sistema difuso generando los valores óptimos de las funciones de membresía, y de esta manera mejorando el rendimiento del mismo. Con los resultados obtenidos se puede determinar cual de los métodos elegidos es el mejor para la optimización de ese controlador en específico.

Abstract

Fuzzy logic is based on decision making with uncertainty, where it is applicable to any system, regardless of whether or not there is a mathematical model, since it is very flexible in terms of reasoning based on human thought. It is being used in basic appliances such as a washing machine, to cutting-edge technology such as in cars.

In soft computing there are also combinations to perform powerful hybridizations, which help to generate a better performance in their tasks, such as the use of bio-inspired algorithms for the optimization of fuzzy controllers, for this reason the present thesis proposes a model competitive multi-metaheuristic for the optimization of fuzzy systems, which determines through experimentation which method is the best for the best efficiency and performance of a controller.

In this case study, a series of algorithms are used, each competitively evaluating the fuzzy system, generating the optimal values of the membership functions, and thus improving its performance. With the obtained results, it can be determined which of the chosen methods is the best for optimizing that specific controller.

Dedication

To my son and Husband, with love, for all your support and understanding.

Acknowledgment

To my husband, my mother and my brothers for their patience and support in everything that I propose.

To Dr. Oscar Castillo, for all his teaching, advice and patience in these years of doctorate, thank you very much.

To my friends and colleagues who have made me laugh, and enjoy every day that we share in the classrooms and laboratory, I will always remember all those beautiful and funny moments in his company Ivette, Oscar, Don Felix and Yunkio thank you.

I thank all the professors for the shared knowledge, Dra. Patricia Melin, Dra. José Soria, Dr. Claudia González, Dr. Fevrier Valdez, Dra. Denisse Hidalgo, Dr. José Mario García Valdés, M.C.C. Alejandra Mancilla, Dra. Gabriela Martínez my admiration and respect.

To the Division of Postgraduate Studies and Research of the Tijuana Technological Institute and the National Council of Science and Technology (CONACYT) for the scholarship awarded to complete my doctorate.

Contents

<i>Abstract</i>	<i>II</i>
<i>Dedication</i>	<i>III</i>
<i>Acknowledgment</i>	<i>IV</i>
Chapter 1. Introduction	- 14 -
Chapter 2. State of the art	- 17 -
2.1 Intelligent computing.....	- 17 -
2.1.1 Fuzzy Logic.....	- 17 -
2.1.2 Type-1 Fuzzy Systems.....	- 18 -
2.1.3 Type-2 Fuzzy Logic.....	- 19 -
Chapter 3. Proposed methodology	- 30 -
3.1 Competitive multi-metaheuristic model.....	- 30 -
Chapter 4. Study cases	- 32 -
4.1 Optimization of a fuzzy controller for autonomous robot navigation.....	- 32 -
4.1.1 Structure and design of the optimization of 3 different fuzzy controller trajectories for an autonomous mobile robot.....	- 32 -
4.1.2 Experimentation and Results.....	- 36 -
4.1.3 Statistical test.....	- 49 -
4.2 Multi-Metaheuristic competitive model for optimization benchmark functions.....	- 55 -
4.2.1 Proposed methodology for optimization of the benchmark functions.....	- 55 -
4.2.2 Results obtained from the optimization of benchmark functions.....	- 58 -
4.3 Optimization of the autonomous robot.....	- 61 -
4.3.1 Optimization of a mobile autonomous robot tracking line.....	- 62 -
4.3.2 Results of the optimization made to the robot autonomous.....	- 65 -
4.4 A new Approach for Dynamic Stochastic Fractal Search with Fuzzy Logic for Parameter Adaptation.....	- 76 -
4.5 Optimization to Dynamic Stochastic Fractal Search with Fuzzy Logic for Parameter Adaptation.....	- 76 -
4.5.1 Results obtained from the optimization.....	- 79 -
4.5.2 Z-test of HFPSO and DSFS Methods.....	- 89 -
4.6 Optimization a type 1 and type 2 fuzzy controller.....	- 92 -
4.6.1 Proposed method for optimizing of a type 1 and type 2 fuzzy logic controller.....	- 92 -
4.6.2 Experimentation and results of the optimization.....	- 103 -
4.6.3 Statistical test.....	- 112 -

Chapter 5. Conclusions - 119 -
References - 120 -
Appendix..... - 124 -

List of figures

Fig. 4.1 Optimization process	- 33 -
Fig. 4.2 Autonomous mobile robot simulation plant.....	- 33 -
Fig. 4.3 Fuzzy inference system of mobile autonomous robot controller	- 34 -
Fig. 4.4 Autonomous mobile robot.....	- 34 -
Fig. 4.5 <i>Desired trajectory 1 of the mobile autonomous robot plant</i>	- 36 -
Fig. 4.6 <i>Desired trajectory, case 2</i>	- 36 -
Fig. 4.7 <i>Desired trajectory 3</i>	- 37 -
Fig. 4.8 <i>Best simulation obtained by FA</i>	- 39 -
Fig. 4.9 <i>Best fuzzy result optimized by SFS</i>	- 40 -
Fig. 4.10 <i>Simulation obtained with the WDO method</i>	- 40 -
Fig. 4.11 <i>Best simulation obtained by DSO</i>	- 41 -
Fig. 4.12 <i>Results obtained for trajectory 1</i>	- 41 -
Fig. 4.13 <i>Best simulation of case 2 by FA</i>	- 43 -
Fig. 4.14 <i>Result of experimentation with WDO</i>	- 44 -
Fig. 4.16 <i>The best result of all experimentation of trajectory 2 by DSO</i>	- 45 -
Fig. 4.17 <i>Results obtained by the competing models in trajectory 2</i>	- 45 -
Fig. 4.18 <i>Best results obtained by DSO</i>	- 47 -
Fig. 4.20 <i>The best result of trajectory 3 by WDO</i>	- 48 -
Fig. 4.21 <i>Results obtained with the FA</i>	- 49 -
Fig. 4.22 <i>Results for trajectory 3</i>	- 49 -
Fig. 4.23 <i>Data flow of the optimization</i>	- 55 -
Fig. 4.24 <i>Desired trajectory, normalized axis</i>	- 62 -
Fig. 4.25 <i>Obtained MSE error</i>	- 63 -
Fig. 4.26. <i>Input ev, not optimized</i>	- 63 -
Fig. 4.27 <i>Input ew, not optimized</i>	- 64 -
Fig. 4.28 <i>Output T1, not optimized</i>	- 64 -
Fig. 4.29 <i>Output T2, not optimized</i>	- 65 -
Fig. 4.30 <i>Input 1 (ev)</i>	- 67 -
Fig. 4.31 <i>Input 2 (ew)</i>	- 68 -
Fig. 4.32 <i>Output 1 (T1)</i>	- 68 -
Fig. 4.33 <i>Output 2 (T2)</i>	- 68 -
Fig. 4.34 <i>Membership functions for Input 1</i>	- 70 -
Fig. 4.35 <i>Membership functions for Input 2</i>	- 70 -
Fig. 4.36 <i>Membership functions for Output 1</i>	- 71 -
Fig. 4.37 <i>Membership functions for Output 2</i>	- 71 -
Fig. 4.38 <i>Input 1 optimized by DSO</i>	- 73 -

Fig. 4.39 Input 2 optimized by DSO	- 73 -
Fig. 4.40 Output 1 optimized by DSO	- 74 -
Fig. 4.41 Output 1 optimized by DSO	- 74 -
Fig. 4.42 Type-1 fuzzy system for controlling of diffusion	- 77 -
Fig. 4.43 Type-1 fuzzy system for controlling of diffusion	- 78 -
Fig. 4.44 DSFS vs HFPSO Results with 10 dimensions	- 88 -
Fig. 4.45 Results of CEC'2017 with 30 dimensions	- 89 -
Fig. 4.46 Proposed optimization method.....	- 93 -
Fig. 4.47 Type-1 fuzzy system for the GSO algorithm	- 94 -
Fig. 4.48 Type-1 fuzzy system for the FA.....	- 94 -
Fig. 4.49 Interval type-2 fuzzy system for the GSO algorithm	- 95 -
Fig. 4.50 Interval type-2 fuzzy system for the FA.....	- 95 -
Fig. 4.51 Structure of individuals in the GSO algorithm and FA.....	- 96 -
Fig. 4.52 Interval type-2 triangular membership function	- 98 -
Fig. 4.53 Interval type-2 trapezoidal membership function	- 99 -
Fig. 4.54 Membership functions of the linear velocity error.....	- 100 -
Fig. 4.55 Membership functions of the angular velocity error.....	- 101 -
Fig. 4.56 Membership functions of Torque 1	- 101 -
Fig. 4.57 Membership functions of the Torque 2.....	- 102 -
Fig. 4.58 The best result (9.64×10^{-02}) obtained with Firefly Algorithm (FA) and its variants-	110 -
Fig. 4.59 The worst result ($9.75 \times 10^{+00}$) obtained with Firefly Algorithm (FA) and its variants	- 111 -
Fig. 4.60 The best result (5.78×10^{-06}) obtain with the GSO algorithm and its variants.-	111 -
Fig. 4.61 The worst result ($7.04 \times 10^{+00}$) obtain with the GSO algorithm and its variants -	112 -

List of tables

Table 4.1 Trajectory 1 results with the FA, WDO, DSO and SFS methods	- 38 -
Table 4.2 Results of 30 experiments generated for trajectory 2	- 42 -
Table 4.3 Results obtained for trajectory 3	- 46 -
Table 4.4 Characteristics of normal distribution z test FA vs WDO on path 1	- 50 -
Table 4.5 Observed and critical values FA vs WDO on path 1	- 50 -
Table 4.6 Characteristics of normal distribution z test WDO vs DSO on path 1.....	- 50 -

Table 4.7	Observed and critical values WDO vs DSO on path 1	50 -
Table 4.8	Characteristics of normal distribution z test WDO vs SFS on path 1	51 -
Table 4.9	Observed and critical values WDO vs SFS on path 1	51 -
Table 4.10	Characteristics of normal distribution z test FA vs WDO on path 2	51 -
Table 4.11	Observed and critical values FA vs WDO on path 2	51 -
Table 4.12	Characteristics of normal distribution z test WDO vs DSO on path 2.....	52 -
Table 4.13	Observed and critical values WDO vs DSO on path 2	52 -
Table 4.14	Characteristics of normal distribution z test WDO vs SFS on path 2.....	52 -
Table 4.15	Observed and critical values WDO vs SFS on path 2.....	52 -
Table 4.16	Characteristics of normal distribution z test FA vs WDO on path 3	53 -
Table 4.17	Observed and critical values FA vs WDO on path 3	53 -
Table 4.18	Characteristics of normal distribution z test WDO vs DSO on path 3.....	53 -
Table 4.19	Observed and critical values WDO vs DSO on path 3	53 -
Table 4.20	Characteristics of normal distribution z test WDO vs SFS on path 3.....	54 -
Table 4.21	Observed and critical values WDO vs SFS on path 3.....	54 -
Table 4.22	<i>Unimodal benchmark functions</i>	56 -
Table 4.23	Multimodal benchmark functions	57 -
Table 4.24	Fixed-dimension multimodal benchmark functions.....	57 -
Table 4.25	Parameters used in the experiments	58 -
Table 4.26	Results for 30 dimensions	58 -
Table 4.27	Parameters used in WDO, DSO and FA the methods.....	59 -
Table 4.28	Results for 64 dimensions	59 -
Table 4.29	Population, Iterations and Dimensions	60 -
Table 4.30	Results obtained with 128 dimensions.....	60 -
Table 4.31	Results for f1 with FA.....	61 -
Table 4.32	Results for f2 with FA.....	61 -
Table 4.33	fuzzy if then rules.....	65 -
Table 4.34	Parameters used in FA, WDO and DSO for fuzzy controller optimization.....	66 -
Table 4.35	Results of the experiments with the fuzzy controller.....	66 -
Table 4.36	Results obtained with WDO.....	68 -
Table 4.37	Results obtained with DSO	72 -
Table 4.38	Summary of results from the optimization methods.....	74 -
Table 4.39	Mean and Deviation Standard.....	75 -
Table 4.40	Parameters statistical test	75 -
Table 4.41	CEC'2017 optimization functions.....	80 -
Table 4.42	Dynamic Stochastic Fractal Search DSFS results with 10 dimensions	81 -
Table 4.43	Dynamic Stochastic Fractal Search DSFS results with 30 dimensions	82 -
Table 4.44	Dynamic Stochastic Fractal Search DSFS results with 100 dimensions	84 -
Table 4.45	<i>HFPSO vs DSFS results with 10 dimensions</i>	85 -
Table 4.46	HFPSO vs DSFS results with 30 dimensions	88 -

Table 4.47 Z-test for 10 dimensions.....	- 90 -
Table 4.48 Results z-test for 30 dimensions	- 91 -
Table 4.49 Fuzzy rules for FGSO	- 96 -
Table 4.50 Fuzzy rules for FA	- 96 -
Table 4.51 Parameters of the GSO and FA algorithms.....	- 103 -
Table 4.52 FA Results.....	- 104 -
Table 4.53 Results for the Fuzzy FA.....	- 105 -
Table 4.54 Results for the Interval Type-2 Fuzzy FA.....	- 106 -
Table 4.55 Results for Galactic Swarm Optimization (GSO).....	- 106 -
Table 4.56 Results for Fuzzy Galactic Swarm Optimization (FGSO1).....	- 107 -
Table 4.57 Results for Interval Type-2 Fuzzy Galactic Swarm Optimization (FGSO IT2) ..	- 108 -
Table 4.58 Best, worst and Average for the FA and its variants.....	- 109 -
Table 4.59 Best, worst and Average for the GSO algorithm and its variants	- 110 -
Table 4.60 Statistical test parameters [61], [62].....	- 113 -
Table 4.61 Results of the Z-test for the GSO algorithm and the FA.....	- 113 -
Table 4.62 Friedman test in comparing GSO versus FA	- 114 -
Table 4.63 Friedman test in comparing FGSO versus FA_T1.....	- 114 -
<i>Table 4.64</i> Friedman test in comparing FGSO IT2 versus FA_T2	- 114 -
Table 4.65 Results of performance the BCO	- 115 -
Table 4.66 MSE results of PSO	- 116 -
Table 4.67 Results of Bat Algorithm	- 116 -
Table 4.68 Summary of results among the proposed methods and other algorithms in the literature	- 117 -

List of equations

(2.1).....	- 19 -
(2.2).....	- 21 -
(2.3).....	- 21 -
(2.4).....	- 22 -
(2.5).....	- 23 -
(2.6).....	- 23 -
(2.7).....	- 24 -
(2.8).....	- 25 -

Chapter 1. Introduction

Currently technology is growing rapidly, every time we can see, read or listen to how new research is emerging to help improve the quality of life of human beings with problem solving. Artificial intelligence [1]–[3] using methods, algorithms and techniques to create robots that have similar characteristics to us. Automatic activities, decision making, detection of images such as artificial neural networks [4]–[6]. Another very interesting part such as expert systems, where fuzzy logic plays a very important role by emulating logical thinking or human reasoning, where its objective is for robots or machines to simulate human behavior.

In addition, we find the meta heuristic algorithms [7]–[9] that are increasingly used as optimization techniques, due to their inspiration in the behavior of biological beings, these methods imitate and / or simulate the strategy of finding the solution to collect food or find a mate for certain animals, exploring and exploiting certain search areas to find the best or optimal solution.

Optimization consists of a set of techniques to find the best solution in a space of possible solutions depending on the problem to be solved. For example, in everyday life most people try to optimize time and money, but there are also other problems in which the need to optimize also arises, such as in the areas of medicine, engineering, robotics, etc. Today meta heuristics and fuzzy logic have been receiving increasing attention as described by the following authors [10], [11].

In this work, an optimization model for membership functions is presented, in which different types of optimization algorithms are used to solve a specific problem. These methods use two essential characteristics of optimization, the exploitation that is a process where the algorithm exploits a certain area in specific to find the global optimum, and the exploration process where each of the element searches in a certain space, limited by the inferior and superior limits. The meta heuristics compete with each other, to demonstrate which has the best performance to find the optimal solution.

The main objective of this thesis is the development of the competitive multi-metahuristic model for the optimization of fuzzy systems, be the core of the model, look for similar works that

help us to make comparisons and with this to validate the results, to publish in magazines and conferences, which would be some of the goals and motivation.

This thesis work is structured by chapters, which are briefly described below:

Chapter 2. State of the art

In Chapter 2, basic concepts are presented that serve to understand the intelligent computing techniques used to make our competitive model; such as the methods used, optimization, fuzzy logic among others.

Chapter 3. Proposed methodology

In chapter 3, provides an explanation of how the proposed model was developed.

Chapter 4. Study cases

Chapter 4, explains the different case studies carried out, where optimizations are made to different problems to be solved.

Chapter 5. Conclusions

This chapter describes the conclusions reached after completing the thesis project.

The publications resulting from this thesis are the following:

Journals:

1.-M. L. Lagunes, et al. *A new Approach for Dynamic Stochastic Fractal Search with Fuzzy Logic for Parameter Adaptation*. Fractal and Fractional, 2021, vol. 5, no 2, p. 33.

2.- M. L. Lagunes, et al. *Optimization of a fuzzy controller for autonomous robot navigation using a new competitive multi- metaheuristic mode*, Soft Computing, 2021, p. 1-20.

3.- E. Bernal, M. L. Lagunes, O. Castillo, O., J. Soria, & F. Valdez. *Optimization of Type-2 Fuzzy Logic Controller Design Using the GSO and FA Algorithms*. International Journal of Fuzzy Systems, 2021, 23(1), 42-57.

- 4.-M. L. Lagunes, O. Castillo, F. Valdez, J. Soria. *Multi-Metaheuristic Competitive Model for Optimization of Fuzzy Controllers*. Algorithms 12(5): 90 (2019)
- 5.- M. L. Lagunes, O. Castillo, J. Soria, M. Garcia, F. Valdez,. *Optimization of granulation for fuzzy controllers of autonomous mobile robots using the Firefly Algorithm*. Granular Computing, 2019, 4(2), 185-195.

Chapters:

- 1.- M. L. Lagunes, O. Castillo, F. Valdez , & J. Soria. *Review of hybrid combinations of metaheuristics for problem solving optimization*. (2021): 221-232.
- 2.- M. L. Lagunes, O. Castillo, F. Valdez, J. Soria, & P. Melin. *Optimization of Fuzzy Controllers for Autonomous Mobile Robots Using the Stochastic Fractal Search Method*. In Recent Advances of Hybrid Intelligent Systems Based on Soft Computing, Springer, 2021, pp. 175-188.
- 3.- M. L. Lagunes, O. Castillo, F. Valdez, J. Soria. *Comparison of Fuzzy Controller Optimization with Dynamic Parameter Adjustment Based on of Type-1 and Type-2 Fuzzy Logic*. Hybrid Intelligent Systems in Control, Pattern Recognition and Medicine, 2020, pp. 47-56.
- 4.- M. L. Lagunes, O. Castillo, J. Soria. *Optimization of Membership Function Parameters for Fuzzy Controllers of an Autonomous Mobile Robot Using the Firefly Algorithm*. Fuzzy Logic Augmentation of Neural and Optimization Algorithms 2018: 199-206.

Conference Proceedings:

- 1.- L. Lagunes Marylu, O. Castillo, F. Valdez, J. Soria. *Comparative Study of Fuzzy Controller Optimization with Dynamic Parameter Adjustment Based on Type 1 and Type 2 Fuzzy Logic*. IFSA/NAFIPS, 2019, pp. 296-305.
- 2.- L. Lagunes Marylu, O. Castillo, F. Valdez, J. Soria , P. Melin. *Parameter Optimization for Membership Functions of Type-2 Fuzzy Controllers for Autonomous Mobile Robots Using the Firefly Algorithm*. NAFIPS, 2018: pp. 569-579.
- 3.- L. Lagunes Marylu, O. Castillo, J. Soria. *Methodology for the Optimization of a Fuzzy Controller Using a Bio-inspired Algorithm*. NAFIPS 2017: 131-137

Chapter 2. State of the art

This chapter describes the concepts used in this thesis work, with which we base ourselves for research, development and experimentation, where the algorithms, intelligent computing methods and each of the tools used for the realization of the model competitive multi-metaheuristic for the optimization of fuzzy systems are illustrated.

2.1 Intelligent computing

Intelligent computing is that branch of artificial intelligence that provides us with techniques for problem solving, such as artificial neural networks that are computational models inspired by the biological neural functioning of the human being, giving the opportunity to create connections to transmit signals that allow the autonomous learning of robots, intelligent applications, etc.

The heuristics and metaheuristics algorithms are methods which are based on the biological inspiration of birds, ants, fireflies among other species to find food or a mate, these algorithms are simulated with mathematical computational models, they allow solving optimization problems. Fuzzy logic provides solutions based on human reasoning, taking into account the insertion that exists between the thinking of each person, creating a system of fuzzy mathematics, this methodology receives input data that does not necessarily have to be mathematical data, but also accepts inaccurate data. In the following citations we will find applications developed with computational intelligence [12], [13].

2.1.1 Fuzzy Logic

Today, fuzzy logic is a computational technique that has given solutions to real problems based on human reasoning, it was developed in (1965) by LA Zadeh [14], [15], where he describes the concepts of fuzzy sets and fuzzy logic. A fuzzy set is a class of objects with a continuum of degrees of membership. This set is characterized by a membership function that assigns each object a membership degree ranging from zero to one. The most used are mentioned below:

1. Triangular membership function.
2. Trapezoidal membership function.
3. Gaussian membership function.
4. Sigmoidal membership function.

2.1.2 Type-1 Fuzzy Systems

The Fuzzy logic is a type of logic that is based on human reasoning, it handles concepts of uncertainty that allow approximate reasoning instead of exact [16], [17]. Its objective is to achieve that a machine has the abilities of a human being, uniting robotics and fuzzy logic, artificial intelligence has given a great advance in science.

This being a power of intelligent computing technique that helps to achieve better quality and efficient work that seemed unattainable, for example in daily use such as cars, computers, washing machines, industrial machines, etc. The logic was developed by Lofti A. Zadeh where he describes the relationship of fuzzy sets used in fuzzy mathematics for the creation of fuzzy systems, with the aim of not needing an exact mathematical model for the solution of a problem, generating mathematical approximations [18].

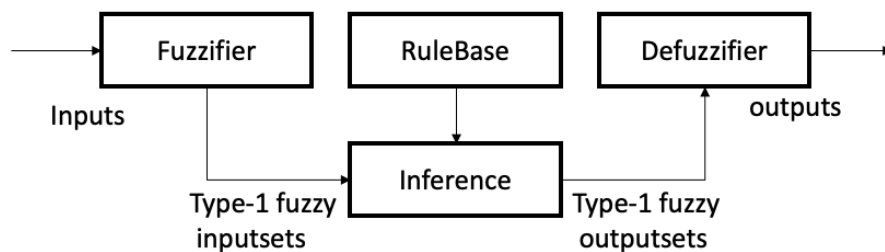


Fig. 2.1 Fuzzy logic system

Figure 2.1 illustrates a fuzzy logic system, it consists of 4 elements: the rule base, the fuzzy inference engine, the diffuser and the defuzzifier. Knowledge is embedded within the rule base in

the form of rules whose antecedent and consequent are fuzzy sets that divide the input and output domains.

2.1.3 Type-2 Fuzzy Logic

The type 2 fuzzy logic is developed as an extension of the type 1 fuzzy logic, using the inserted between the fuzzy systems of [0,1] giving a degree of membership to each of the membership functions of each set, these systems can model nonlinear systems generating better performance than type 1 fuzzy logic. Some of the mathematically modeled membership functions are described below.

The following Equations presents a type 2 membership function, where the primary membership degree is x and the secondary membership degree are the weights of type 1 fuzzy logic.

The FOU is the uncertainty footprint where all the nested membership functions are concentrated, where there is the upper membership function described as (UMF) $\bar{\mu}(x)$ and the lower one (LMF) $\underline{\mu}(x)$, among these are as mentioned before n numbers of type 1 fuzzy sets, forming a type 2 system [19]–[23].

$$\begin{aligned}
 \tilde{\mu}(x) &= [\underline{\mu}(x), \bar{\mu}(x)] = \text{itrapatype2}(x, [a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, \alpha]) \\
 &\text{where } a_1 < a_2, b_1 < b_2, c_1 < c_2, d_1 < d_2 \\
 \mu_1(x) &= \max\left(\min\left(\frac{x - a_1}{b_1 - a_1}, 1, \frac{d_1 - x}{d_1 - c_1}\right), 0\right) \\
 \mu_2(x) &= \max\left(\min\left(\frac{x - a_2}{b_2 - a_2}, 1, \frac{d_2 - x}{d_2 - c_2}\right), 0\right) \\
 \bar{\mu}(x) &= \begin{cases} \max(\mu_1(x), \mu_2(x)) & \forall x \notin (b_1, c_2) \\ 1 & \forall x \in (b_1, c_2) \end{cases} \\
 \underline{\mu}(x) &= \min\left(\alpha, \min(\mu_1(x), \mu_2(x))\right)
 \end{aligned} \tag{2.1}$$

$$\begin{aligned} \tilde{\mu}(x) &= [\underline{\mu}(x), \bar{\mu}(x)] = \text{igaussatype2}(x, [\sigma, m, \alpha]) \\ \underline{\mu}(x) &= \alpha \exp \left[-\frac{1}{2} \left(\frac{x-m}{\sigma} \right)^2 \right] \text{ Where } 0 < \alpha < 1 \\ \bar{\mu}(x) &= \exp \left[-\frac{1}{2} \left(\frac{x-m}{\sigma} \right)^2 \right] \end{aligned} \quad (2.2)$$

The Equations 2.1 and 2.2 shows the mathematical structure of the trapezoidal and Gaussian membership functions with type 2 fuzzy logic, have twice the number of parameters compared to type 1 fuzzy logic.

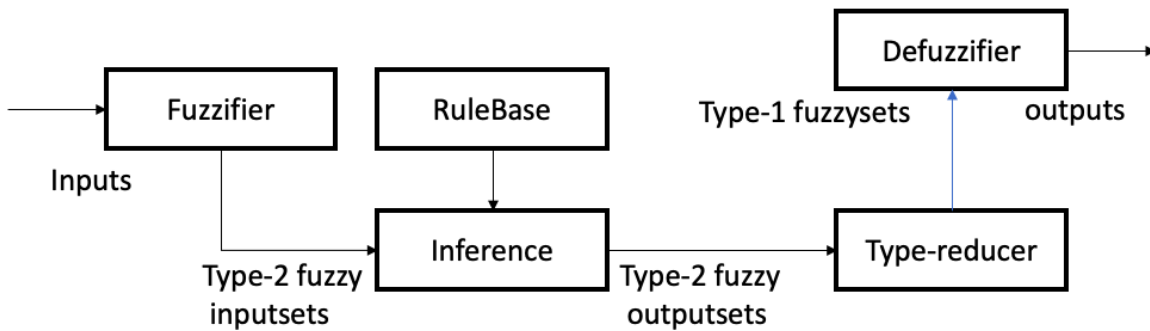


Fig. 2.2 Type 2 fuzzy logic system

A type 2 fuzzy logic system is similar to its type 1 counterpart, the main difference is that at least one of the fuzzy sets is type 2 and a type reducer is needed to convert the type 2 fuzzy out sets in type 1sets so that it can be processed by the defuzzifier to give an output as illustrates in Figure 2.2. General type 2 are computationally intensive because type-reduction is very intensive.

2.1.3.1 Meta-heuristics

Metaheuristics are algorithms based on behaviors in most cases of biological beings, each of these algorithms has the ability to solve complex problem, depending on the area and the

problem to be solved is how the method is chosen, these are being used widely to solve analytical problems in which the traditional algorithms were not functional [24], [25], [26] for example in non-linear problems, control problems, optimization problems, etc. The metaheuristics used in the proposed model are described below.

2.1.3.2 Firefly Algorithm (FA)

Developed in 2008 by X. S. Yang [27], [28] the FA based on the inspiration of the behavior of flickering fireflies, which emit a light to attract a mate or prey, the brighter a firefly shines, the more it attracts others. The method has three important rules.

- a) All fireflies are unisex therefore any firefly can be attracted to another.
- b) The firefly that shines the most is the one that attracts the other fireflies, and the more distance there is between them, the more its attraction decreases.
- c) The brightness of the firefly is given by the value of its objective function.

The method is represented by the following equations:

The equation for defining *attraction* is:

$$\beta = \beta_0 e^{-\gamma r^2} \quad (2.2)$$

Where:

β_0 is the initial attraction in $r = 0$ and has values in $[0,1]$, e is represented by 2,71828 is the basis of the Natural logarithms, r is the distance between two fireflies and γ determines the variation of attractiveness by increasing the distance between the fireflies and has values in $[0,1]$.

The equation for defining the *movement* is:

$$x_i^{t+1} = x_i^t + e^{-\gamma r_{ij}^2} (x_j^t - x_i^t) + \alpha_t \varepsilon_i^t \quad (2.3)$$

Where:

x_i^{t+1} represents the next position, x_i^t is the actual position $e^{-\gamma r_{ij}^2} (x_j^t - x_i^t)$ is the attraction between fireflies, α_t represent the randomization with α being the parameter of randomness $\alpha \in [0,1]$ and finally the vector of random numbers extracted from a Gaussian distribution is ε_i^t .

The equation for defining the *distance* is:

$$r_{ij} = \sqrt{\sum_{k=1}^d (x_{i,k} - x_{j,k})^2} \quad (2.4)$$

where r_{ij} is Euclidean distance between two fireflies i and j , $x_{i,k}$ represents the k th component of the spatial coordinate, x_i is the i -th firefly and d is number of dimensions.

Begin

Objective function $f(x), x = (x_1, \dots, x_d)^T$

Generate initial population of n fireflies $x_i, i = 1, 2, \dots, n$

Formulate light intensity I so that it is associated with $f(x)$

While ($t < \text{MaxGeneration}$)

Define absorption coefficient γ

for $i = 1 : n$ (n fireflies)

for $j = 1 : n$ (n fireflies)

if ($I_j > I_i$),

move firefly i towards j

end if

Vary attractiveness with distance r via $\exp(-\gamma r^2)$

Evaluate new solutions and update light intensity

end for j

end for i

Rank the fireflies and find the current best

end while

Post-processing the results and visualization

end

The Figure shows the pseudo-code of firefly algorithm, at the beginning the method declares the objective function, initializes the population, the intensity variable is determined by the objective function, then the number of generations, absorption coefficient, the lights move towards the

brightest, the distance is determined by r , then are evaluated and ranked, obtaining the one with the best performance.

2.1.3.3 Wind Driven Optimization (WDO)

The optimization methods mostly have two important characteristics: exploration and exploitation, the most popular is the PSO and ACO [29], [30], which is inspired by the behavior of swarms of animals, such as fish and birds that work as a team to hunt. In addition to PSO, other metaheuristics are also inspired by colonies of swarms, for example, WDO [31], [32], which has a behavior very similar to PSO because it also has a swarm, but not of fish or birds, as it has a swarm of air particles that also update their position and speed, and travel through the atmosphere.

The WDO is represented by two main equations as follows.

The equation for defining the *new speed* is:

$$v_{t+1}^i = (1 - \sigma)v_t^i - gx_t^i \left(RT \left| \frac{1}{r} - 1 \right| (x_{opt} - x_t^i) \right) + \left(\frac{cu_t^{other\ dim}}{r} \right) \quad (2.5)$$

Where:

the maximum pressure is equal to 1, α represents the coefficient of friction, current speed is given by v_t^i , g is the gravitational acceleration, the actual position is equal to x_t^i , the R and T represent the universal gas constant and temperature respectively, r is the Air package range, where all air packages are sorted in descending order by their pressure, x_{opt} represents the best global position and c is the Coriolis constant.

The equation for defining the *Next position* is:

$$x_{t+1}^i = x_t^i + v_{t+1}^i \quad (2.6)$$

```

Begin
Initialize parameters of WDO algorithm ( $\alpha, g, RT, c$ )
Randomize velocity and position of each dimension
For iteration = n
    Call objective function,
    Rank the air parcels,
    Store the best objective function value and its dimensions,
    Update velocity of air parcel and update position
Stopping criterion
Display optimal parameters
end

```

The Figure shows the pseudo-code of wind driven optimization, in the method is define the cost function, population size and boundary condition, randomize position and velocity, evaluate pressure value for each air parcel, update velocity and position, check stopping criterion and end.

2.1.3.4 Drone Squadron Optimization (DSO)

Of the algorithms not inspired by biological beings, we find the DSO [33], [34] which is a method that is inspired by artifacts, such as drones. Its behavior is based on autonomous robots that fly over the search area, guided by software, where the necessary information is programmed so that they can go out in search of the optimal value of an objective function, and it is semi-adaptive because the software of each drone is update whenever necessary.

The DSO has two important rules:

- 1.- Maintain partial control of the search.
- 2.- To develop new control software for drones.

Based on these important rules, the two main equations are presented.

The equation for defining the *Perturbation* is:

$$P = Departure + Offset \quad (2.7)$$

$$TC = calculate(P) \quad (2.8)$$

Departure is a set of coordinates (a solution point in the search space), Offset is a function that returns the actual perturbation movement (a numerical value), and P is the complete perturbation equation that has to be calculated to return the trial coordinates.

The equation for defining the *Violation* is:

$$Violation_{team} = \sum_{j=1}^D \left\{ |TmC - UB_j| + |LB_j - TmC| \right\} \quad (2.10)$$

where *TmC* represents the team coordinates, *N* is the number of drones per team, the dimension is represented by *D*, the upper limit and lower limit are represented by *UB_j* and *LB_j* respectively.

The Figure shows the pseudocode of the drone square optimization, the drone team and parameters, positions are initialized, all the drones are evaluated, the drone with the best coordinates is selected, the new position is assigned to all the drones updating the software.

```

1: begin
2: Initialize firmware of each team and parameters;
3: Initialize drones' positions;
4: Evaluate all the drones (calculate their fitness values) and record the best one  $P_{best}$ ;
5: Select the N best drones to be  $CBP$ , corresponding to fitness  $CBF$ 
6: while termination criterion is not satisfied do
7:   for  $i=1$  to 2 do
8:     Generate  $Position_i$  according to firmware
9:     Choose the method of recombination and apply it to  $Position_i$  and  $CBP$ ,
       resulting in  $Position_i$ 
10:    Choose the method of boundary-correction and apply it to  $Position_i$  and  $CBP$ ,
       resulting in  $Position_i$ 
11:    Evaluate each  $Position_i$ , resulting in  $Fit_i$ 
12:   end for
13:   for  $i=1$  to  $N$  do
14:     compare  $Fit_{1,i}, Fit_{2,i}$ , record  $LBP_i = Position_{bestD,i}$ ,  $LBF_i = Fit_{bestD,i}$ 
15:     if  $Stag = Stagnation$ 
16:       If  $(LBP_i < CBF_i)$ 
17:         then  $CBP_j = LBP_i, CBF_j = LBF_i$ 
18:       else if  $(U(0,1) < Pacc)$ 
19:         then  $CBP_j = LBP_i, CBF_j = LBF_i$ 
20:     else
21:       If  $(LBP_i < CBF_i)$ 
22:         then  $CBP_j = LBP_i, CBF_j = LBF_i$ 
23:   end for
24:   if  $Stag = Stagnation$ 
25:     then  $Stag=0; ConStag++$ 
26:   if improve
27:     then  $Stag=0; ConStag=0$ 
28:   if  $ConStag = ContinuousStagnation$ 
29:     then restart
30:   else  $Stag++$ 
31:   Update the firmware
32:   Record  $P_{best}$ 
33: end while
34: return  $P_{best}$ 
35: end

```

2.1.3.5 Stochastic Fractal Search (SFS)

The algorithms PSO [35], GA [36], DE [37], ACO [38], etc., are widely used, but recently other methods are also gaining relevance and being used in various areas such as the SFS method [39], [40] inspired by fractals where a pattern is formed randomly which grows in a similar way, starting with a point called a seed and continuing with continuous seeds that grow around the initial one.

The SFS has the following equations.

$$P = LB + \varepsilon(UP - LB) \quad (2.11)$$

The population of particles P is randomly generated based on the constraints of the problem after setting the lower limit LB and the upper limit UB.

The equation defining the *Diffusion* process (exploitation) is expressed as follows:

$$GW_1 = Gaussian(\mu_{BP}, \sigma) + (\varepsilon BP - \varepsilon' P_i) \quad (2.12)$$

$$GW_2 = Gaussian(\mu_p, \sigma) \quad (2.13)$$

where , ε' represent random numbers in the range [0,1], BP is the best position of the point, i -th point in the group is represented by P_i and *Gaussian* is a function that generates a random number from a normal distribution with a mean μ parameter and a standard deviation parameter σ :

$$\sigma = \frac{\log g}{g} |P_i - BP| \quad (2.14)$$

Where:

$\frac{\log g}{g}$ represents the approach to zero as the generation number g increases.

The equation defining the *Update Processes* (exploration) is:

$$Pa_i = \frac{rank(P_i)}{N} \quad (2.15)$$

Where:

N is the number of particles in the group and Pa_i is the computed probability value for a particle whose rank among the others is given by the “rank” function. The particles are classified per their fitness value and then a probability value is given to each particle i .

$$P'_i(j) = P_x(j) - (P_y(j) - P_i(j)) \quad (2.16)$$

Where:

the modified component is represented by $P'_i(j)$, and P_x, P_y are different points randomly chosen from the group. P'_i replaces P_i if it has a better fitness value.

Begin

Initialize parameters of SFS algorithm (iteration, population)

Diffusion process

Set maximum number of diffusion parameters

Gaussian walk

Apply first Gaussian walk

Find the new position of particles

Apply second Gaussian walk

Find the second new position of particles

Updating process

Ranked all search agent

Use two updating process

Find best search agent

End

The Figure shows the pseudocode of the Stochastic Fractal Search, initializes the parameters of the method, the number of diffusions is established, the first Gaussian walk is applied for the new position of particles, the second Gaussian walk is applied for the second new position of particles, the update process where the particles are ranked and the best one is found.

Chapter 3. Proposed methodology

Different types of methods have been developed for optimization, these algorithms are used individually, jointly or hybrid, these techniques used are analyzed and developed depending on the performance of the method in the problem to be solved. For example the author [41] made a toolbox for the optimization of benchmark functions, where he uses two algorithms PSO and GA, other authors such as [43]–[47] use different metaheuristics for optimizations.

A continuation in this chapter describes the development of a competitive multi-metaheuristic model for the optimization of fuzzy systems, to obtain the best algorithm to solve a specific problem.

3.1 Competitive multi-metaheuristic model

The methodology proposes a competitive multi-metaheuristic model for the optimization of fuzzy systems as illustrated in the Fig. 3.1 it is described below:

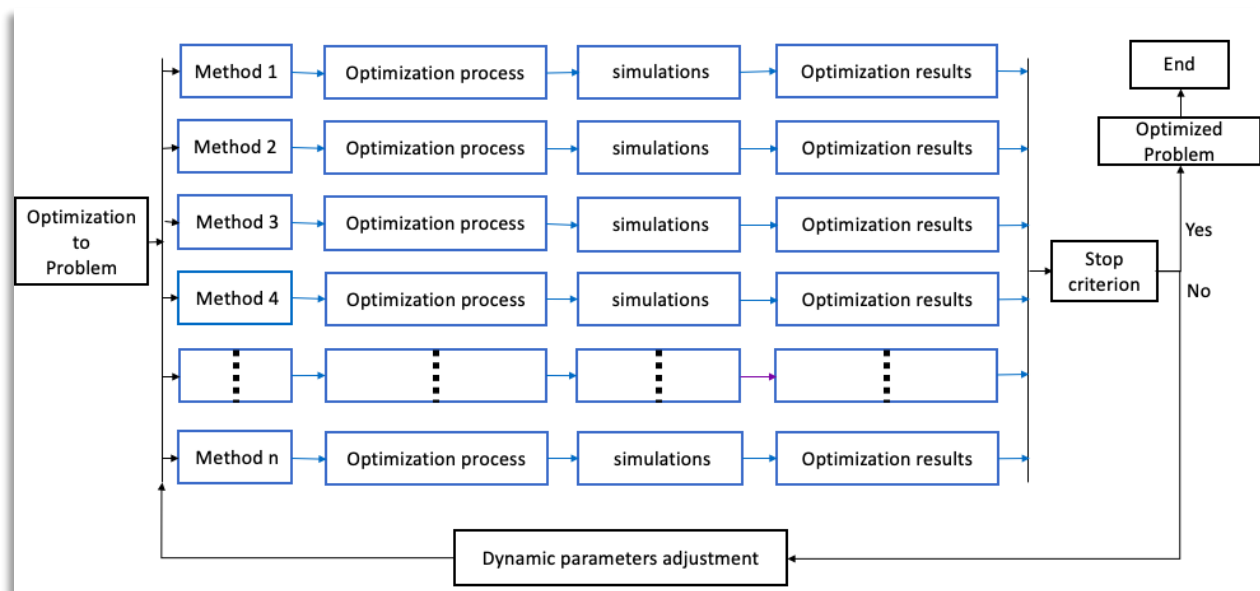


Fig. 3.1 Competitive multi-metaheuristic model for optimization of fuzzy systems

Module 1 Problem to be optimized:

It corresponds to the input, where the problem to be optimized is defined.

Module 2 Method: The problem is established in each of the optimization methods (the methods are up to n, depending on the optimization problem), these algorithms are chosen after a long search in the state of the art, where they are assigned based on their performance with related work or motivation to experiment with new methods.

Module 3 optimization process: The optimization process begins in each of the algorithms to find the best solution. Each solution given by the metaheuristics is simulated to determine the performance of the methods, using some established estimator, to measure the difference between the expected value and the actual value found.

Module 4 simulations: The optimization is simulated.

Module 5 Optimization results: The best result of each of the methods given by the simulation is obtained.

Module 6 Stop criterion: The results are evaluated with a metric or iterations.

Module 7: If any of the results meet the stopping criteria, it is established that the optimization problem is solved or optimized.

If none of the methods meets the stopping criterion, then the dynamic parameter adjustment is applied to the method that showed the worst performance in the optimization, in this way the algorithm once again has the opportunity to compete with the other metaheuristics.

Depending on each problem to be solved, different algorithms can be used in the model, that is, the methods that in the literature proved to be efficient in similar cases are sought, the number of meta-heuristics can also vary, they are added or removed depending on the difficulty of the problem.

Chapter 4. Study cases

In this chapter the optimizations made with the proposed model are described, the comparisons developed with the algorithms chosen for optimization are also analyzed, analyzing the performance of each one and determining with which one obtains a better result of the specific optimization problem, these experiments are organized as case studies.

4.1 Optimization of a fuzzy controller for autonomous robot navigation

In this first study case, the optimization of a fuzzy system is carried out for the simulation of 3 different trajectories of a mobile autonomous robot, generating the values of the parameters of the optimal membership functions, as explained below:

4.1.1 Structure and design of the optimization of 3 different fuzzy controller trajectories for an autonomous mobile robot

In Fig. 4.1 the methodology used in the development of the optimization for the parameters of the membership functions is illustrated, it starts with the input, in this case the fuzzy controller, then it is sent to each of the proposed methods, once you have the optimized fuzzy system, it is simulated in the robot plant to know the effectiveness of the optimization, the result of the simulation is evaluated by means of a stop criterion that can be the iterations or some other metric. In this case, the method that meets the stop criteria is the winner. In other words, the method that generated the best data vector that optimized the fuzzy controller and therefore the best solution to the given problem, if the stop criterion is not met, then the method that produced the worst result is improved and starts the competition again.

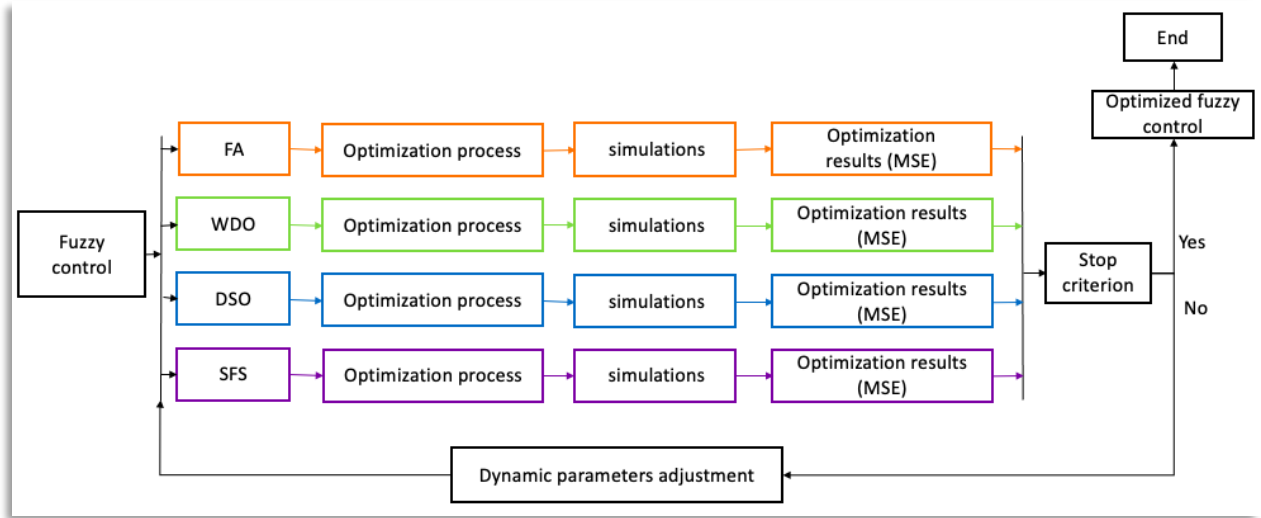


Fig. 4.1 Optimization process

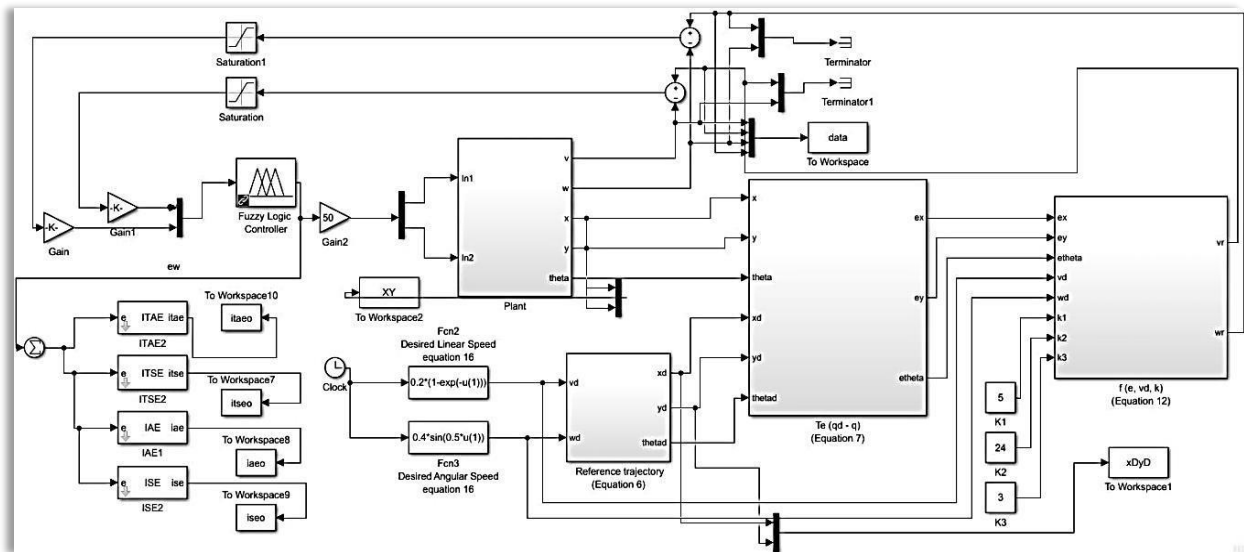


Fig. 4.2 Autonomous mobile robot simulation plant

As can be seen in Fig. 4.2 the components of the autonomous robot plant, the objective is to achieve the minimum distance between the real trajectory generated by the robot and the desired one already established, which in this case is defined by sine and cosine equations. To achieve this objective, a multi-metaheuristic model is used in which each of the individual metaheuristics used evaluates the fuzzy controller, optimizing the parameters of the membership functions until the method that best suits each of the experimentation paths is found. The fuzzy controller [47] has two inputs, the linear and angular velocities, which provide the information to control the robot. The inputs have two trapezoidal and one triangular membership functions, the output corresponds

to the turning movement made by the rear tires, and these are represented as pair 1 (t1) and pair 2 (t2) respectively, and the outputs have 3 triangular members as illustrated in Fig. 4.3.

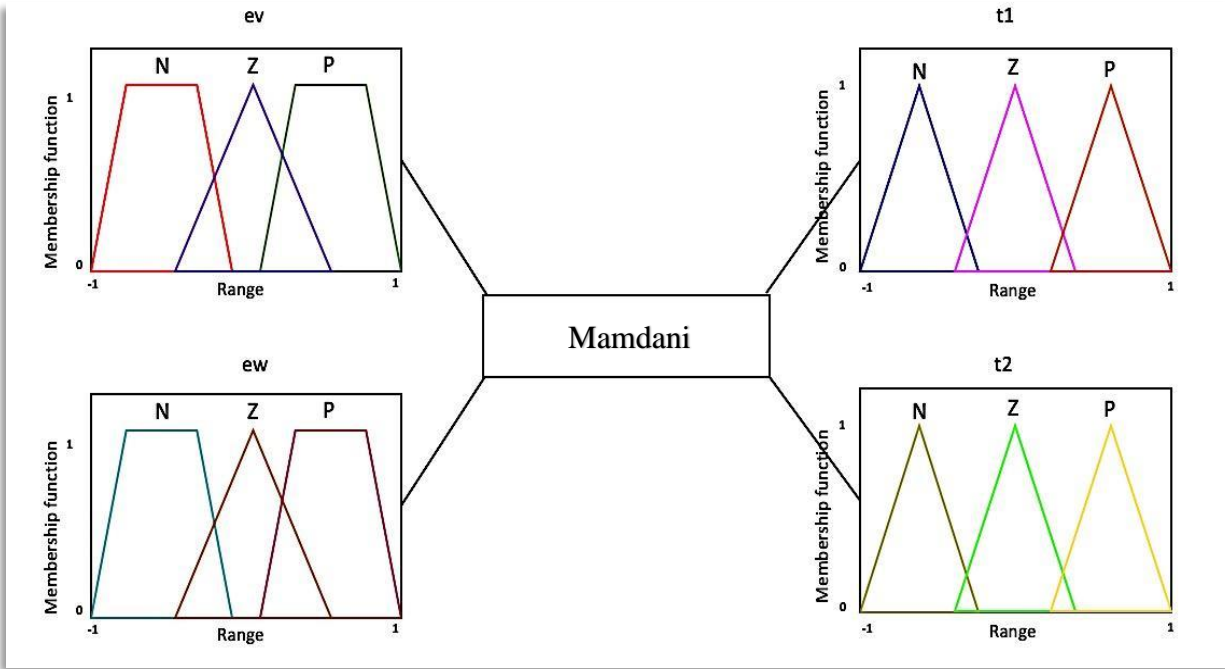


Fig. 4.3 Fuzzy inference system of mobile autonomous robot controller

The autonomous mobile robot [48] is made up of two driving wheels located on the same axis, and a crazy wheel in the front for the balance of the robot, as illustrated in Fig. 4.4.

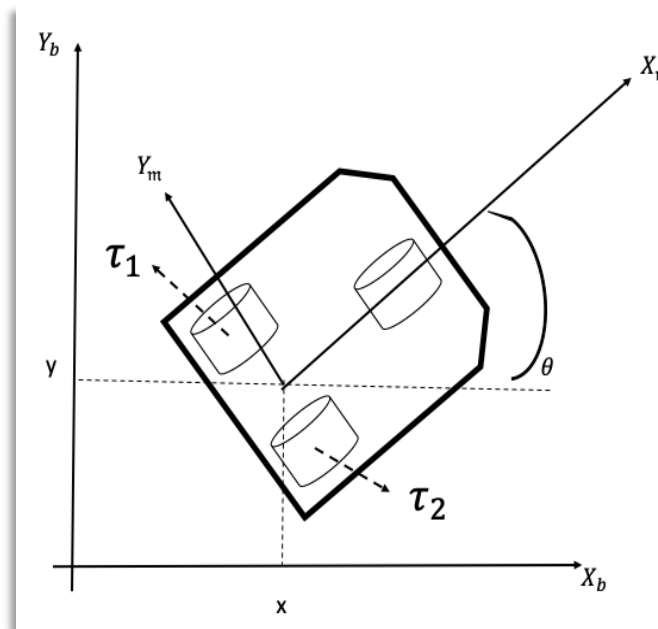


Fig. 4.4 Autonomous mobile robot

The free wheel is not considered in this model, as shown in the following equation 4.1:

$$M(q)\dot{v} + C(q, \dot{q})v + Dv = \tau + P(t) \quad (4.1)$$

where,

$q = (x, y, \theta)^T$ is the vector of the configuration coordinates,

$v = (v, w)^T$ is the vector of velocities,

$\tau = (\tau_1, \tau_2)$ is the vector of torques applied to the wheels of the robot where τ_1 and τ_2 denote the torques of the right and left wheels,

$P \in R^2$ is the uniformly bounded disturbance vector,

$M(q) \in R^{2 \times 2}$ is the positive-definite inertia matrix,

$C(q, \dot{q})v$ is the vector of centripetal and Coriolis forces, and

$D \in R^{2 \times 2}$ is a diagonal positive-definite damping matrix.

The kinematic system is represented by Equation 4.2:

$$\dot{q} = \begin{bmatrix} \cos \theta & 0 \\ \sin \theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \quad (4.2)$$

where:

(x, y) is the position in the X-Y reference frame,

θ is the angle between the heading direction and the x-axis,

v and w are the linear and angular velocities.

Furthermore, the following equation shows the non-holonomic constraint that this system has, which corresponds to a non-slip wheel condition preventing the robot from moving sideways.

$$\dot{y} \cos \theta - \dot{x} \sin \theta = 0 \quad (4.3)$$

4.1.2 Experimentation and Results

The proposed methodology was tested with 3 different trajectories, as can be seen in Fig. 4.5 the trajectory 1 is formed by Equation 4.4, path 2 illustrated in Fig. 4.6 and is defined by Equation 4.5 and finally, Equation 4.6 forms trajectory 3 that shown in Fig. 4.7.

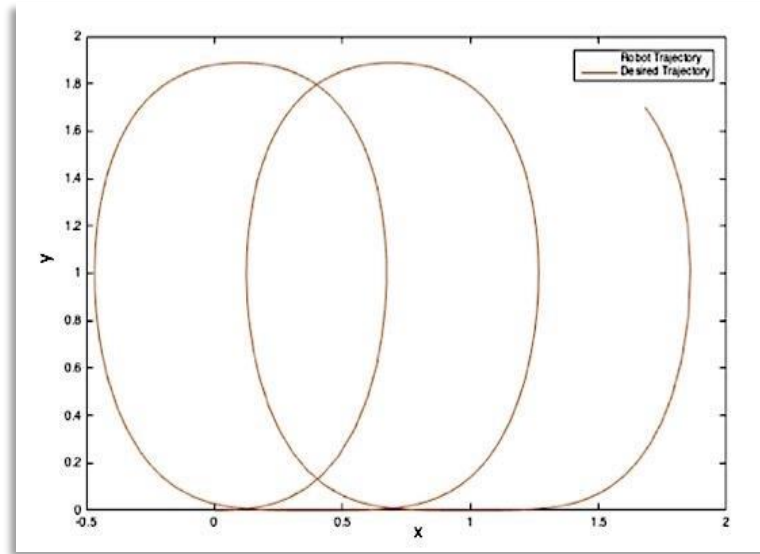


Fig. 4.5 Desired trajectory 1 of the mobile autonomous robot plant

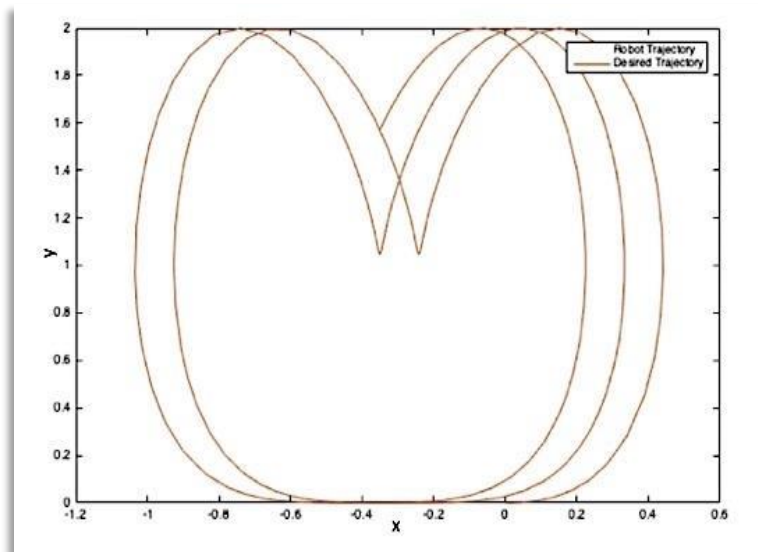


Fig. 4.6 Desired trajectory, case 2

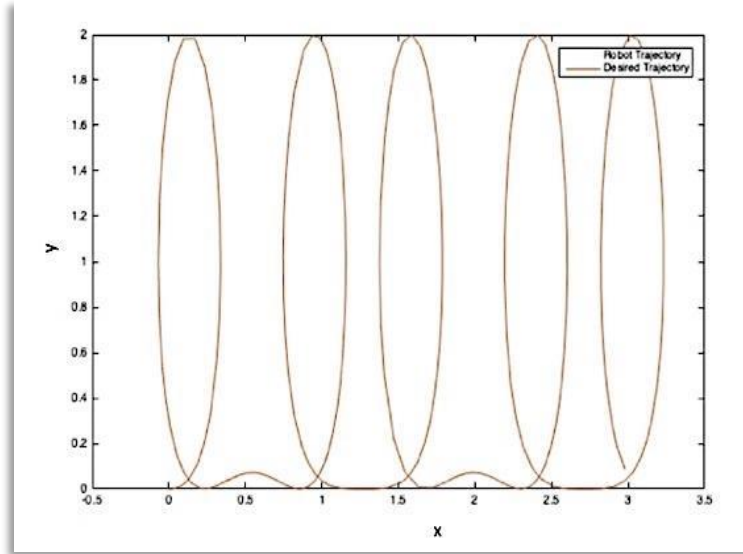


Fig. 4.7 Desired trajectory 3

$$0.4 \sin(0.3 u(1)) \quad (4.4)$$

$$0.7 \sin(0.3 u(1)) \quad (4.5)$$

$$1 \sin(0.3 u(1)) \quad (4.6)$$

The error is defined by Equation 4.7, which is of the Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y})^2 \quad (4.7)$$

where \hat{Y} is a vector of n predictions, and Y is the vector of actual values.

As can be noted, from Table 4.1, in column 2, the obtained values by the FA are much dispersed; the method generated a minimum value of 9.63×10^{-01} and a maximum value $8.91 \times 10^{+00}$. Column 3, corresponding to the results of the WDO method, shows values dispersed between $x10^{-01}$ to $x10^{-04}$, where it can be noted that the adaptation behavior of the WDO method to the problem of trajectory 1, was generating good parameter values for the membership functions, although they are not constant. The values shown in column 4 are obtained by the DSO method in the case of trajectory 1, where it is noted that the algorithm obtained good results in the

optimization of the fuzzy controller generating the best MSE of 6.83×10^{-05} . The SFS method achieved a good adaptation to problem 1, as illustrated in column 5. The presented results are very constant and this demonstrates the effectiveness of the algorithm for the solution of this specific optimization problem.

Table 4.1 Trajectory 1 results with the FA, WDO, DSO and SFS methods

Experiment	MSE FA	MSE WDO	MSE DSO	MSE SFS
1	9.63 $\times 10^{-01}$	1.15×10^{-03}	1.01×10^{-01}	1.16×10^{-05}
2	$1.37 \times 10^{+00}$	1.09 $\times 10^{-04}$	9.64×10^{-02}	1.17×10^{-05}
3	$4.46 \times 10^{+00}$	7.68 $\times 10^{-04}$	2.19×10^{-02}	2.33×10^{-05}
4	$6.81 \times 10^{+00}$	2.60×10^{-03}	4.39×10^{-03}	7.47 $\times 10^{-06}$
5	$8.91 \times 10^{+00}$	4.26 $\times 10^{-04}$	2.40×10^{-03}	5.29×10^{-05}
6	$2.66 \times 10^{+00}$	2.97×10^{-02}	2.40×10^{-03}	8.89 $\times 10^{-06}$
7	5.56 $\times 10^{-01}$	1.31×10^{-03}	2.40×10^{-03}	2.26×10^{-05}
8	$2.06 \times 10^{+00}$	1.44×10^{-03}	2.40×10^{-03}	8.38 $\times 10^{-06}$
9	$2.27 \times 10^{+00}$	7.36 $\times 10^{-04}$	5.68×10^{-04}	1.74×10^{-05}
10	$2.46 \times 10^{+00}$	2.60 $\times 10^{-04}$	1.33×10^{-04}	2.27×10^{-05}
11	$2.66 \times 10^{+00}$	9.89×10^{-03}	1.25×10^{-04}	1.75×10^{-05}
12	$2.92 \times 10^{+00}$	2.73×10^{-03}	1.22×10^{-04}	2.80×10^{-06}
13	$2.69 \times 10^{+00}$	1.08×10^{-03}	1.22×10^{-04}	4.33×10^{-05}
14	$2.46 \times 10^{+00}$	2.05×10^{-03}	8.98 $\times 10^{-05}$	3.48×10^{-05}
15	$1.29 \times 10^{+00}$	1.06×10^{-02}	8.98 $\times 10^{-05}$	3.19 $\times 10^{-06}$
16	$2.98 \times 10^{+00}$	4.96×10^{-03}	2.15×10^{-03}	2.17×10^{-04}
17	$3.38 \times 10^{+00}$	5.20 $\times 10^{-04}$	8.28 $\times 10^{-05}$	3.42×10^{-06}
18	$3.67 \times 10^{+00}$	2.06×10^{-03}	8.28 $\times 10^{-05}$	3.06×10^{-05}
19	$3.56 \times 10^{+00}$	7.14 $\times 10^{-04}$	7.98 $\times 10^{-05}$	4.70 $\times 10^{-06}$
20	$4.00 \times 10^{+00}$	1.79×10^{-03}	4.59×10^{-01}	1.37×10^{-04}
21	$3.34 \times 10^{+00}$	2.29 $\times 10^{-04}$	7.73 $\times 10^{-05}$	1.65×10^{-05}
22	$4.75 \times 10^{+00}$	1.44×10^{-03}	7.73 $\times 10^{-05}$	5.42×10^{-05}
23	$2.34 \times 10^{+00}$	7.94 $\times 10^{-04}$	3.91×10^{-04}	3.17 $\times 10^{-06}$

24	$4.37 \times 10^{+00}$	6.32×10^{-02}	7.73×10^{-05}	1.17×10^{-04}
25	$4.68 \times 10^{+00}$	3.80×10^{-04}	7.73×10^{-05}	2.95×10^{-05}
26	$1.63 \times 10^{+00}$	1.25×10^{-04}	3.46×10^{-04}	1.87×10^{-05}
27	$4.95 \times 10^{+00}$	1.99×10^{-03}	3.45×10^{-04}	3.05×10^{-05}
28	$1.99 \times 10^{+00}$	1.73×10^{-03}	7.73×10^{-05}	2.73×10^{-05}
29	$2.69 \times 10^{+00}$	4.74×10^{-03}	3.44×10^{-04}	2.10×10^{-05}
30	$2.73 \times 10^{+00}$	1.07×10^{-03}	6.83×10^{-05}	1.26×10^{-06}

As can be noted in Fig. 4.8, Fig. 4.9, Fig. 4.10 and Fig. 4.11, the simulation results of the best fuzzy controllers generated by the competing methods are presented, respectively. The best method in adapting to the optimization problem of a fuzzy controller in trajectory 1 was SFS because it generates the best results when compared to the other methods, as illustrated in Fig. 4.12.

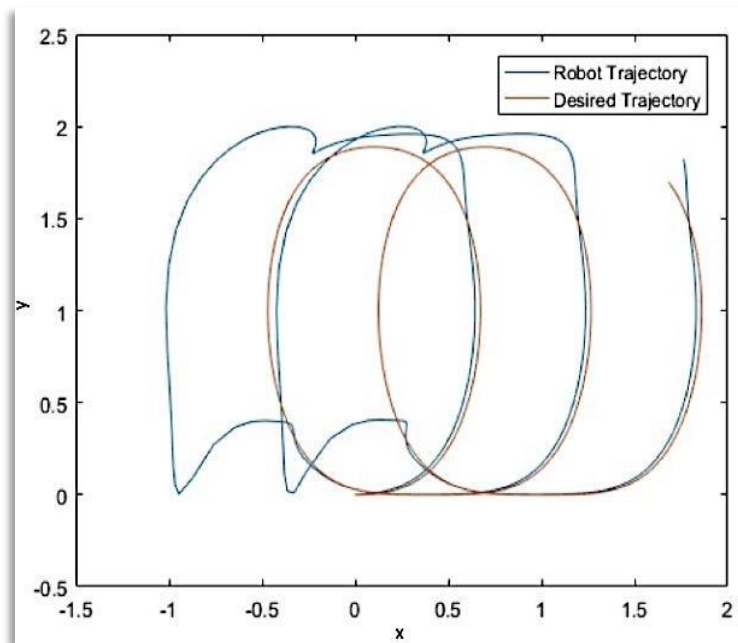


Fig. 4.8 Best simulation obtained by FA

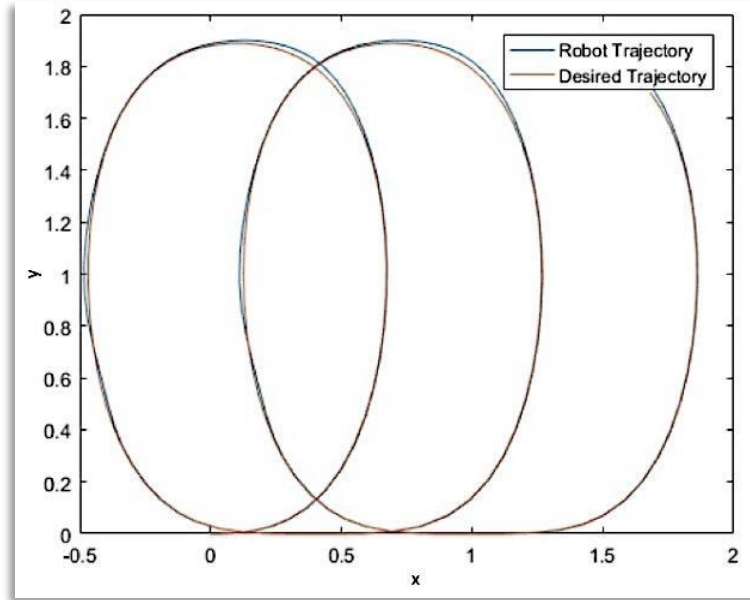


Fig. 4.9 Best fuzzy result optimized by SFS

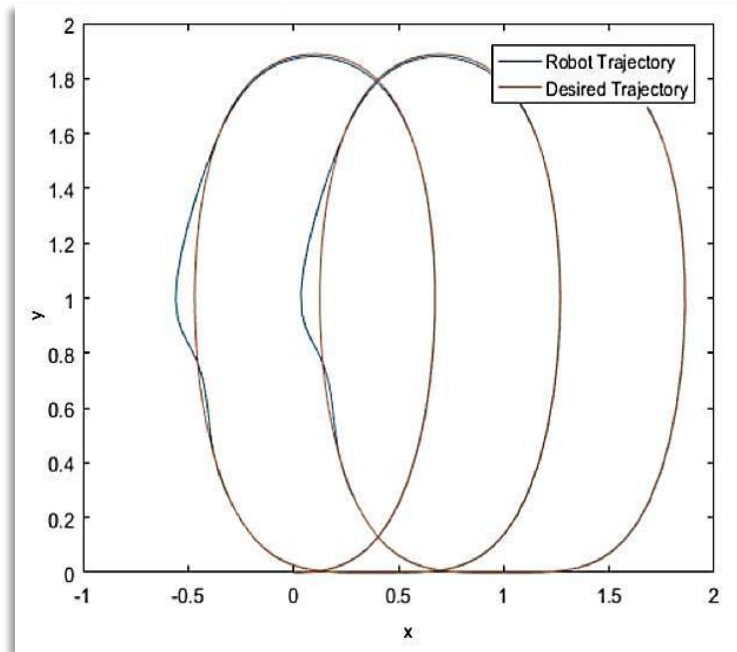


Fig. 4.10 Simulation obtained with the WDO method

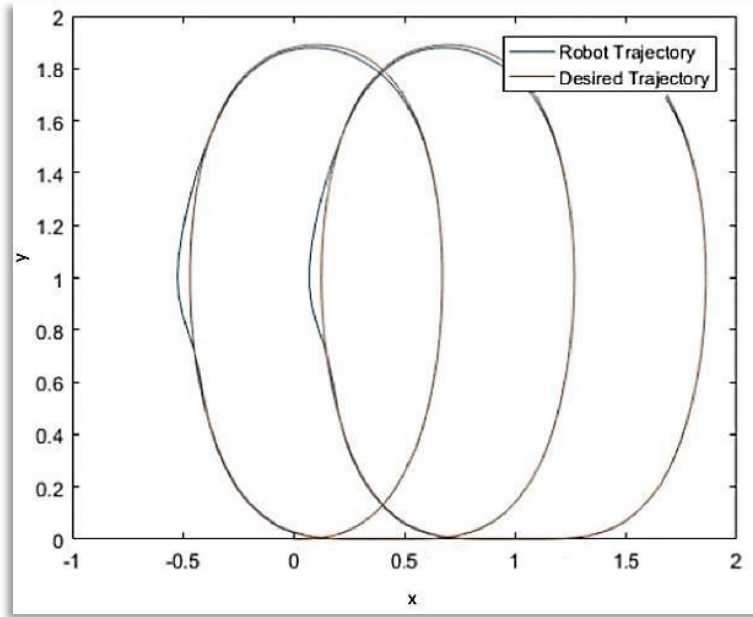


Fig. 4.11 Best simulation obtained by DSO

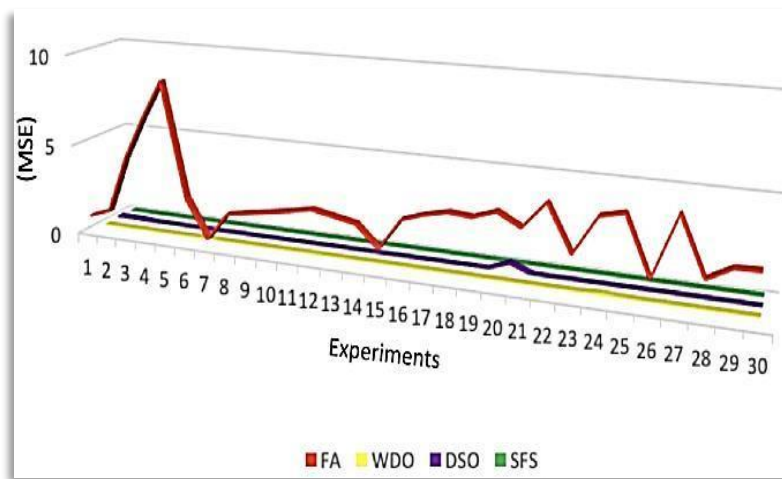


Fig. 4.12 Results obtained for trajectory 1

As was observed in the previous case, the FA method was also the least flexible according to the results presented in Table 4.2. Where it is shown that the MSE is very high compared to the DSO, WDO and SFS, methods.

The values found by the WDO method are not the ideal ones, but with this model and the tested cases, it is shown that the method has flexibility in the optimization problem, generating a MSE of 6.51×10^{-05} , as shown in column 3.

As can be noted in column 4, the DSO method suffers from stagnation at $x10^{-04}$, since most of the MSE numbers are at that same value, there is not much dispersion of the data, so it is concluded that at that value the global minimum is reached, except for two minimum values at $1.25x10^{-06}$. Column 5 illustrates the results obtained with the SFS method for trajectory 2, where the dispersion of data is observed for each of the 30 performed experiments.

Fig. 4.13, Fig. 3.14, Fig. 4.15 and Fig. 4.16 show the best simulations generated by each of the competing methods, from which it can be stated that, despite being a complex trajectory, that the DSO method was the one that obtained the best optimized fuzzy controller. Fig. 4.17 shows the comparison of the results of the methods.

Table 4.2 Results of 30 experiments generated for trajectory 2

Experiment	FA	WDO	DSO	SFS
1	$1.72x10^{+00}$	$5.48x10^{-04}$	$1.01x10^{-01}$	$2.20x10^{-02}$
2	$2.87x10^{+00}$	$2.76x10^{-03}$	$9.64x10^{-02}$	$9.85x10^{-03}$
3	$2.90x10^{+00}$	$3.07x10^{-04}$	$2.19x10^{-02}$	$1.31x10^{-02}$
4	$3.36x10^{+00}$	$2.81x10^{-03}$	$4.39x10^{-03}$	$2.67x10^{-02}$
5	$5.11x10^{+00}$	$7.55x10^{-05}$	$2.40x10^{-03}$	$4.39x10^{-03}$
6	$1.43x10^{+00}$	$2.68x10^{-03}$	$2.40x10^{-03}$	$9.87x10^{-03}$
7	$2.88x10^{+00}$	$7.82x10^{-04}$	$2.40x10^{-03}$	$6.83x10^{-03}$
8	$1.40x10^{+00}$	$1.55x10^{-04}$	$2.40x10^{-03}$	$3.69x10^{-02}$
9	$1.24x10^{+00}$	$6.76x10^{-04}$	$5.68x10^{-04}$	$1.88x10^{-02}$
10	$4.16x10^{+00}$	$2.50x10^{-03}$	$1.33x10^{-04}$	$3.76x10^{-02}$
11	$5.97x10^{-01}$	$2.00x10^{-03}$	$1.25x10^{-04}$	$2.38x10^{-02}$
12	$9.87x10^{-01}$	$5.46x10^{-04}$	$1.22x10^{-04}$	$1.95x10^{-02}$
13	$1.15x10^{+00}$	$2.58x10^{-04}$	$1.22x10^{-04}$	$4.61x10^{-03}$
14	$5.60x10^{-01}$	$6.51x10^{-05}$	$8.98x10^{-05}$	$4.23x10^{-03}$
15	$1.69x10^{+00}$	$2.94x10^{-03}$	$8.98x10^{-05}$	$5.67x10^{-02}$
16	$1.27x10^{+00}$	$1.71x10^{-03}$	$2.15x10^{-03}$	$2.03x10^{-02}$
17	$3.34x10^{+00}$	$1.18x10^{-03}$	$8.28x10^{-05}$	$4.01x10^{-02}$
18	$1.19x10^{+00}$	$1.53x10^{-03}$	$8.28x10^{-05}$	$6.37x10^{-02}$
19	$1.84x10^{+00}$	$2.46x10^{-04}$	$7.98x10^{-05}$	$2.85x10^{-03}$

20	$3.34 \times 10^{+00}$	1.40×10^{-03}	4.59×10^{-01}	6.47×10^{-02}
21	6.48×10^{-01}	2.85×10^{-04}	7.73×10^{-05}	9.42×10^{-02}
22	$1.92 \times 10^{+00}$	1.30×10^{-02}	7.73×10^{-05}	2.67×10^{-02}
23	6.05×10^{-01}	5.10×10^{-04}	3.91×10^{-04}	3.98×10^{-03}
24	3.70×10^{-01}	2.53×10^{-04}	7.73×10^{-05}	2.17×10^{-03}
25	9.44×10^{-01}	2.53×10^{-03}	7.73×10^{-05}	3.26×10^{-03}
26	$4.55 \times 10^{+00}$	9.67×10^{-04}	3.46×10^{-04}	4.04×10^{-03}
27	$2.79 \times 10^{+00}$	7.54×10^{-04}	3.45×10^{-04}	6.97×10^{-02}
28	$2.76 \times 10^{+00}$	1.95×10^{-04}	7.73×10^{-05}	1.31×10^{-02}
29	$3.92 \times 10^{+00}$	3.21×10^{-03}	3.44×10^{-04}	4.21×10^{-03}
30	$3.39 \times 10^{+00}$	5.71×10^{-03}	6.83×10^{-05}	3.22×10^{-02}

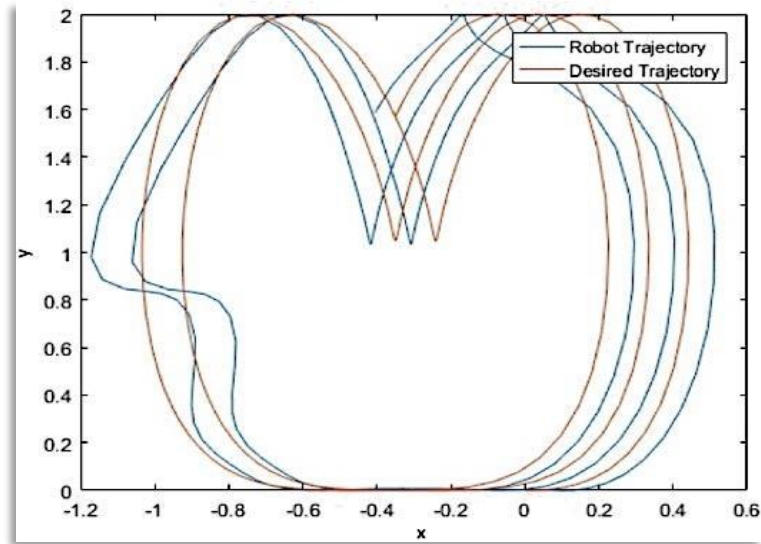


Fig. 4.13 Best simulation of case 2 by FA

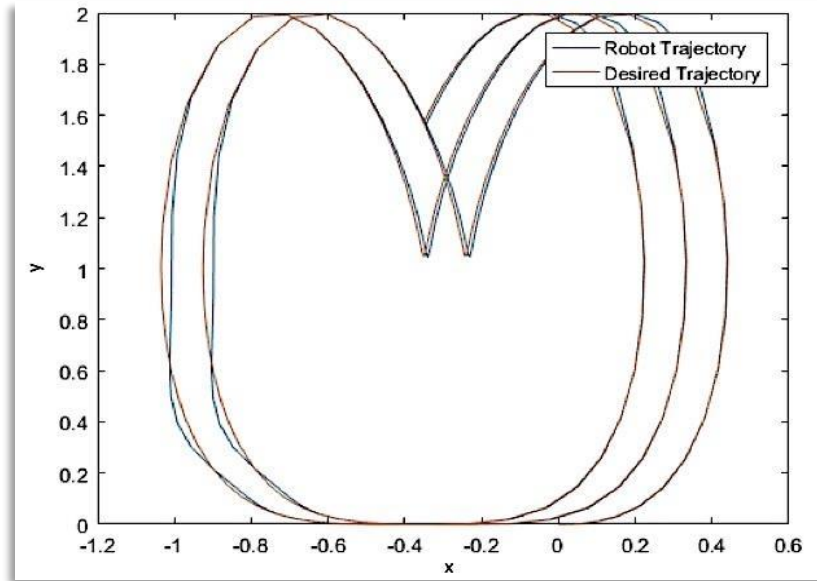


Fig. 4.14 Result of experimentation with WDO

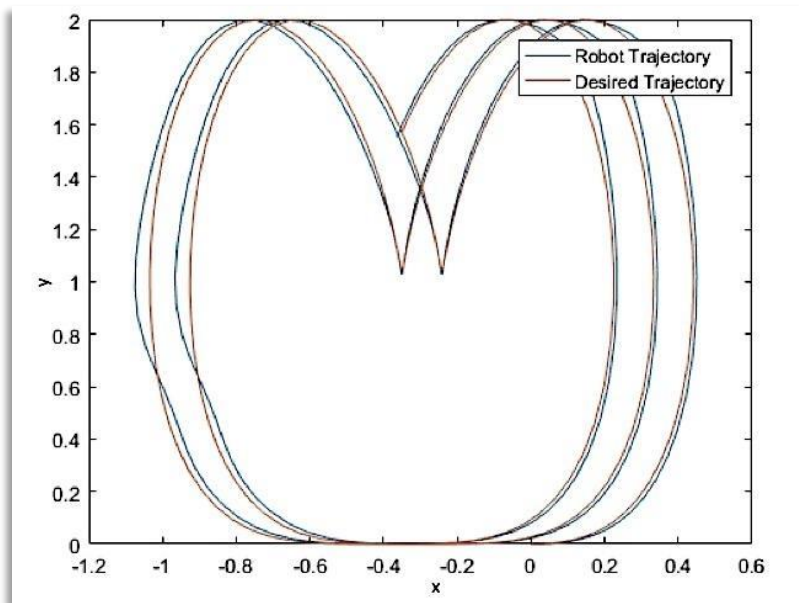


Fig. 4.15 Result of trajectory 2 with SFS

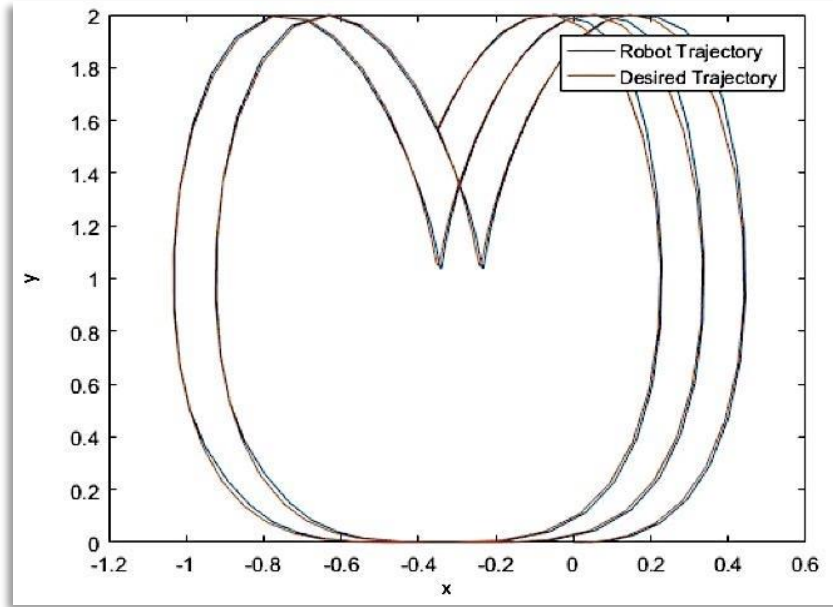


Fig. 4.16 The best result of all experimentation of trajectory 2 by DSO

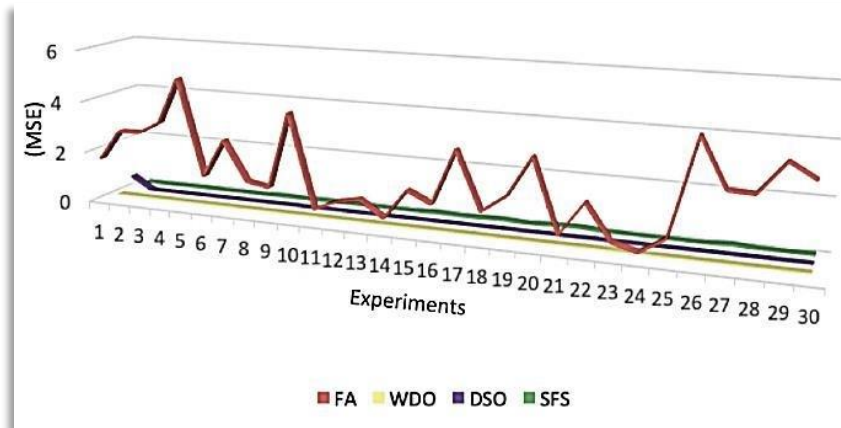


Fig. 4.17 Results obtained by the competing models in trajectory 2

The results obtained by the FA method for the optimization of trajectory 3 are described below in Table 4.3. Column 3 shows the results generated by the WDO method, where the best MSE is 1.14×10^{-05} . The results of 30 experiments carried out by the DSO method are illustrated in column 4, where a significant variation in error is observed, and with this data dispersion it can be stated that for the method it was hard to be able to maintain a low MSE.

Although the MSE values are very different as noted in column 5, the SFS was the method that delivered the best results in the optimization of trajectory 3. In Fig. 4.18, Fig. 4.19, Fig. 4.20 and Fig. 4.21 we show the best simulation results of the optimization of trajectory 3, where it is observed that WDO and SFS were the ones that generated a better data vector that delivered an optimized fuzzy controller generating an acceptable MSE for this problem.

Table 4.3 Results obtained for trajectory 3

Experiment	FA	WDO	DSO	SFS
1	$2.07x10^{+00}$	$1.78x10^{-04}$	$2.12x10^{-02}$	$1.70x10^{-04}$
2	$7.71x10^{-01}$	$1.41x10^{-03}$	$1.84x10^{-06}$	$1.60x10^{-05}$
3	$4.80x10^{+00}$	$3.15x10^{-04}$	$3.40x10^{-04}$	$9.75x10^{-05}$
4	$1.48x10^{+00}$	$2.21x10^{-03}$	$1.70x10^{-03}$	$4.59x10^{-05}$
5	$1.10x10^{+00}$	$1.10x10^{-03}$	$1.02x10^{-02}$	$2.18x10^{-04}$
6	$9.89x10^{-01}$	$4.95x10^{-03}$	$1.06x10^{-06}$	$8.81x10^{-05}$
7	$4.23x10^{+00}$	$1.04x10^{-03}$	$5.21x10^{-01}$	$1.16x10^{-05}$
8	$3.85x10^{+00}$	$7.99x10^{-05}$	$2.65x10^{-04}$	$1.17x10^{-05}$
9	$5.29x10^{+00}$	$7.72x10^{-04}$	$1.97x10^{-04}$	$2.33x10^{-05}$
10	$2.61x10^{+00}$	$2.06x10^{-04}$	$8.07x10^{-03}$	$7.47x10^{-06}$
11	$7.13x10^{-01}$	$1.68E-03$	$9.70x10^{-06}$	$5.29x10^{-05}$
12	$4.00x10^{+00}$	$2.23x10^{-04}$	$9.22x10^{-06}$	$8.89x10^{-06}$
13	$3.84x10^{+00}$	$1.14x10^{-05}$	$1.02x10^{-06}$	$2.26x10^{-05}$
14	$4.79x10^{+00}$	$3.55x10^{-03}$	$2.21x10^{-04}$	$8.38x10^{-06}$
15	$2.02x10^{+00}$	$3.06x10^{-03}$	$1.96x10^{-04}$	$1.74x10^{-05}$
16	$1.96x10^{+00}$	$5.88x10^{-04}$	$5.05x10^{-04}$	$2.27x10^{-05}$
17	$3.84x10^{+00}$	$1.25x10^{-03}$	$2.65x10^{-04}$	$1.75x10^{-05}$
18	$2.76x10^{+00}$	$1.50x10^{-04}$	$4.64x10^{-05}$	$2.80x10^{-06}$
19	$2.48x10^{+00}$	$9.94x10^{-04}$	$1.97x10^{-04}$	$4.33x10^{-05}$
20	$5.23x10^{+00}$	$4.09x10^{-04}$	$2.65x10^{-04}$	$3.48x10^{-05}$
21	$3.64x10^{+00}$	$6.42x10^{-04}$	$1.75x10^{-06}$	$3.19x10^{-06}$
22	$2.44x10^{+00}$	$3.88x10^{-04}$	$2.07x10^{-04}$	$2.17x10^{-04}$

23	$4.83 \times 10^{+00}$	7.56×10^{-03}	2.04×10^{-04}	3.42×10^{-06}
24	$3.87 \times 10^{+00}$	1.33×10^{-03}	1.04×10^{-06}	3.06×10^{-05}
25	$2.64 \times 10^{+00}$	9.29×10^{-04}	1.84×10^{-06}	4.70×10^{-06}
26	$5.00 \times 10^{+00}$	2.40×10^{-03}	2.65×10^{-04}	1.37×10^{-04}
27	$2.97 \times 10^{+00}$	2.17×10^{-04}	1.97×10^{-04}	1.80×10^{-05}
28	$5.49 \times 10^{+00}$	2.27×10^{-04}	1.86×10^{-03}	9.95×10^{-05}
29	$5.76 \times 10^{+00}$	1.44×10^{-03}	3.50×10^{-04}	4.79×10^{-05}
30	1.30×10^{-01}	1.06×10^{-03}	2.01×10^{-04}	2.98×10^{-04}

In Fig. 4.22 we show a comparison of the results of trajectory 3 with the four competing methods

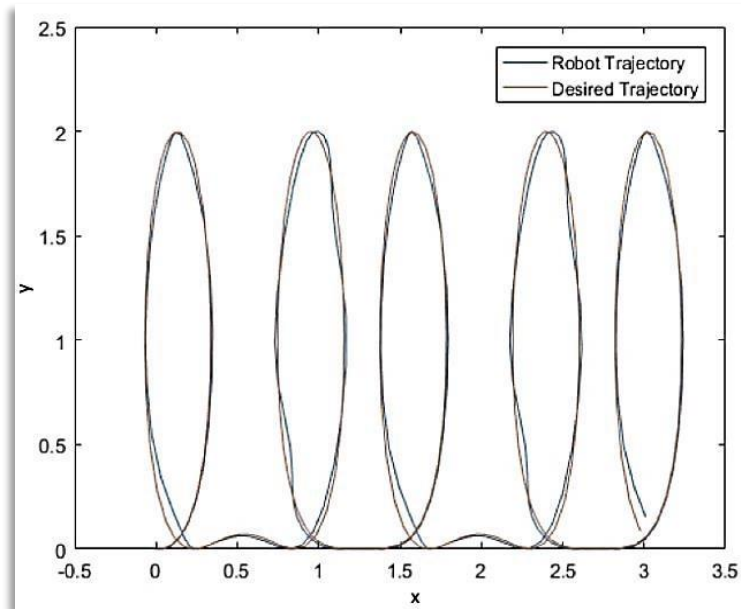


Fig. 4.18 Best results obtained by DSO.

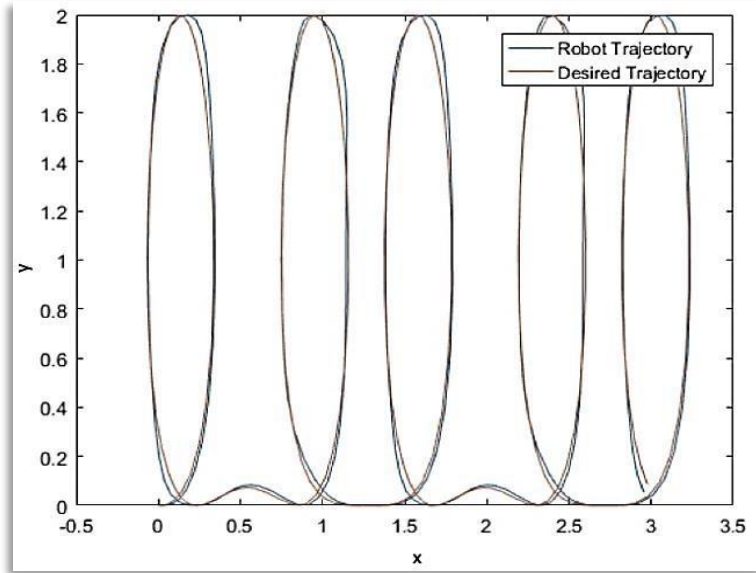


Fig. 4.19 Trajectory optimized by SFS.

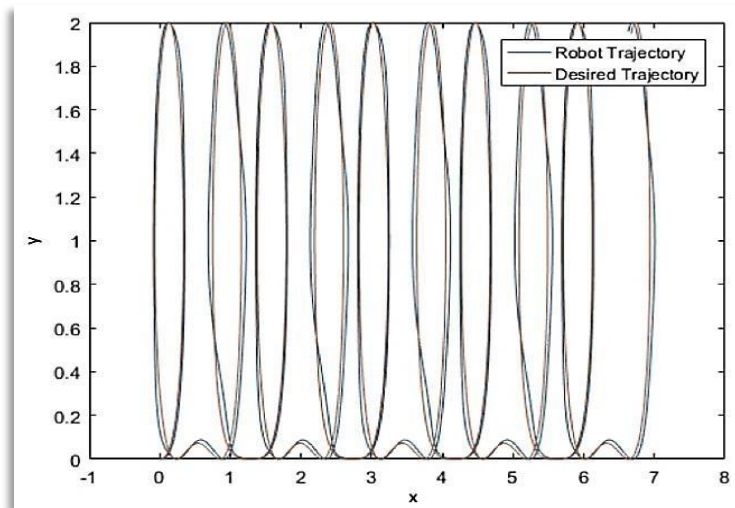


Fig. 4.20 The best result of trajectory 3 by WDO.

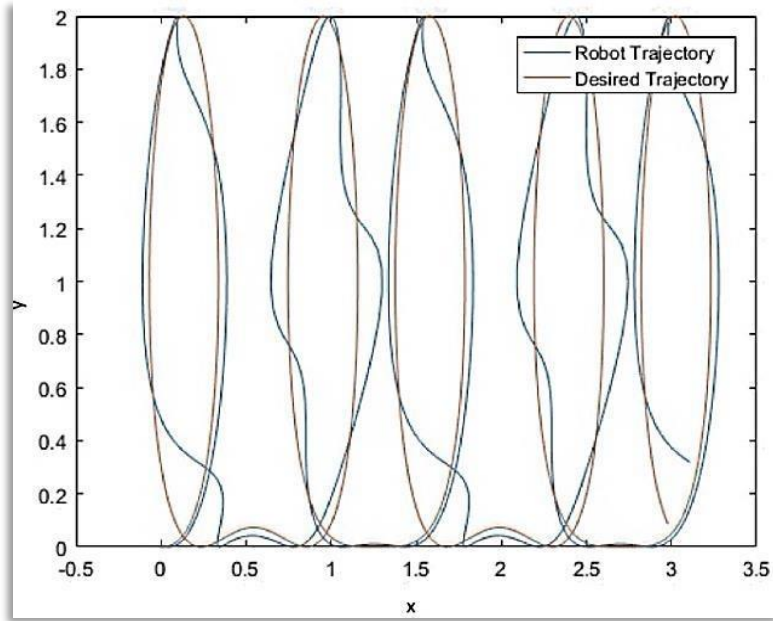


Fig. 4.21 Results obtained with the FA.

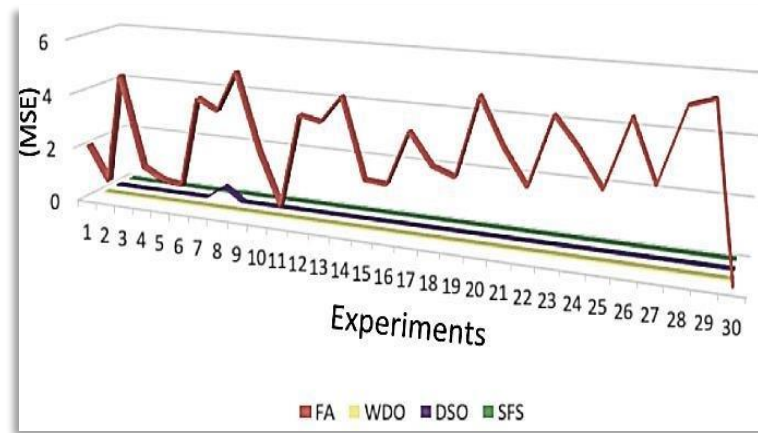


Fig. 4.22 Results for trajectory 3

4.1.3 Statistical test

As previously explained, three different trajectories or different cases were experimented with, using the MSE as an error metric. To perform the hypothesis test for comparing the results of the four methods, the analysis of ANOVA variance was performed for each of the cases.

This part shows z test, for each of the trajectories, where each of the methods were compared one by one:

Table 4.4 Characteristics of normal distribution z test FA vs WDO on path 1

Variable	Observations	Minimum	Maximum	Averages	Standard deviations
FA	30	0.556	8.910	3.187	1.719
WDO	30	0.000	0.063	0.005	0.012

Table 4.5 Observed and critical values FA vs WDO on path 1

Difference	3.182
Z (observed value)	10.139
Z (critical value)	-1.645
Value p (Unilateral)	1.000
α	0.05

Table 4.6 Characteristics of normal distribution z test WDO vs DSO on path 1

Variable	Observaciones	Minimum	Maximum	Averages	Standard deviations
WDO	30	0.000	0.063	0.005	0.012
DSO	30	0.000	0.459	0.023	0.086

Table 4.7 Observed and critical values WDO vs DSO on path 1

Difference	-0.018
Z (observed value)	-1.150
Z (critical value)	-1.645
Value p (Unilateral)	0.125
α	0.05

Table 4.8 Characteristics of normal distribution z test WDO vs SFS on path 1

Variable	Observations	Minimum	Maximum	Averages	Standard deviations
WDO	30	0.000	0.063	0.005	0.012
SFS	30	0.000	0.000	0.000	0.000

Table 4.9 Observed and critical values WDO vs SFS on path 1

Difference	0.005
Z (observed value)	2.211
Z (critical value)	-1.645
Value p (Unilateral)	0.986
α	0.05

Table 4.10 Characteristics of normal distribution z test FA vs WDO on path 2

Variable	Observations	Minimum	Maximum	Averages	Standard deviations
FA	30	0.556	8.910	3.187	1.719
WDO	30	0.000	0.063	0.005	0.012

Table 4.11 Observed and critical values FA vs WDO on path 2

Difference	3.182
Z (observed value)	10.139
Z (critical value)	-1.645
Value p (Unilateral)	1.000
α	0.05

Table 4.12 Characteristics of normal distribution z test WDO vs DSO on path 2

Variable	Observations			Averages	Standard deviations
		Minimum	Maximum		
WDO	30	0.000	0.063	0.005	0.012
DSO	30	0.000	0.459	0.023	0.086

Table 4.13 Observed and critical values WDO vs DSO on path 2

Difference	-0.018
Z (observed value)	-1.150
Z (critical value)	-1.645
Value p (Unilateral)	0.125
α	0.05

Table 4.14 Characteristics of normal distribution z test WDO vs SFS on path 2

Variable	Observations	Minimum	Maximum	Averages	Standard deviations
SFS	30	0.000	0.000	0.000	0.000

Table 4.15 Observed and critical values WDO vs SFS on path 2

Difference	0.005
Z (observed value)	2.211
Z (critical value)	-1.645

Value p (Unilateral)	0.986
α	0.05

Table 4.16 Characteristics of normal distribution z test FA vs WDO on path 3

Variable	Observations	Minimum	Maximum	Averages	Standard deviations
FA	30	0.130	5.760	3.186	1.606
WDO	30	0.000	0.008	0.001	0.002

Table 4.17 Observed and critical values FA vs WDO on path 3

Difference	3.185
Z (observed value)	10.860
Z (critical value)	-1.645
Value p (Unilateral)	1.000
α	0.05

Table 4.18 Characteristics of normal distribution z test WDO vs DSO on path 3

Variable	Observations	Minimum	Maximum	Averages	Standard deviations
WDO	30	0.000	0.008	0.001	0.002
DSO	30	0.000	0.521	0.019	0.095

Table 4.19 Observed and critical values WDO vs DSO on path 3

Difference	-0.018
Z (observed value)	-1.015

Z (critical value)	-1.645
Value p (Unilateral)	0.155
α	0.05

Table 4.20 Characteristics of normal distribution z test WDO vs SFS on path 3

Variable	Observations	Minimum	Maximum	Averages	Standard deviations
WDO	30	0.000	0.008	0.001	0.002
SFS	30	0.000	0.000	0.000	0.000

Table 4.21 Observed and critical values WDO vs SFS on path 3

Difference	0.001
Z (observed value)	4.307
Z (critical value)	-1.645
Value p (Unilateral)	1.000
α	0.05

Tables 4.13 to 4.31 show the results obtained with the normal distribution z test, the observed and critical values with an estimate with a value of 0.05. Tested with 3 different trajectories, the optimization was performed to find the best values of the parameters of the membership functions

of the tracking fuzzy controller for an autonomous mobile robot, 30 experiments were carried out and a comparative study was developed with algorithms proposed in the model.

4.2 Multi-Metaheuristic competitive model for optimization benchmark functions

For this second case study, the optimization Unimodal, multimodal and Fixed-dimension multimodal benchmark functions is performed as described below [49]:

4.2.1 Proposed methodology for optimization of the benchmark functions

As described previously in chapter 3, the methodology consists of creating a competitive model based on a set of metaheuristics, the general idea of this proposal is to have a series of optimization methods, which receive an input (specific problem) for being processed, and thus optimized, the method that produces a better result than the others in the competition, will demonstrate that it is the best to optimize that problem, and a metric is used to evaluate the particular results of the problem.

Figure 2 shows the proposed optimization data flow, it starts with input, in this case the problem that will be optimized, then goes to the methods as an objective function, continuing with the optimization process, continues with the results obtained, then the comparison with the stop criterion, if necessary, the algorithm that gave the worst result is adjusted and the cycle starts again, and if not, the problem is considered optimized.

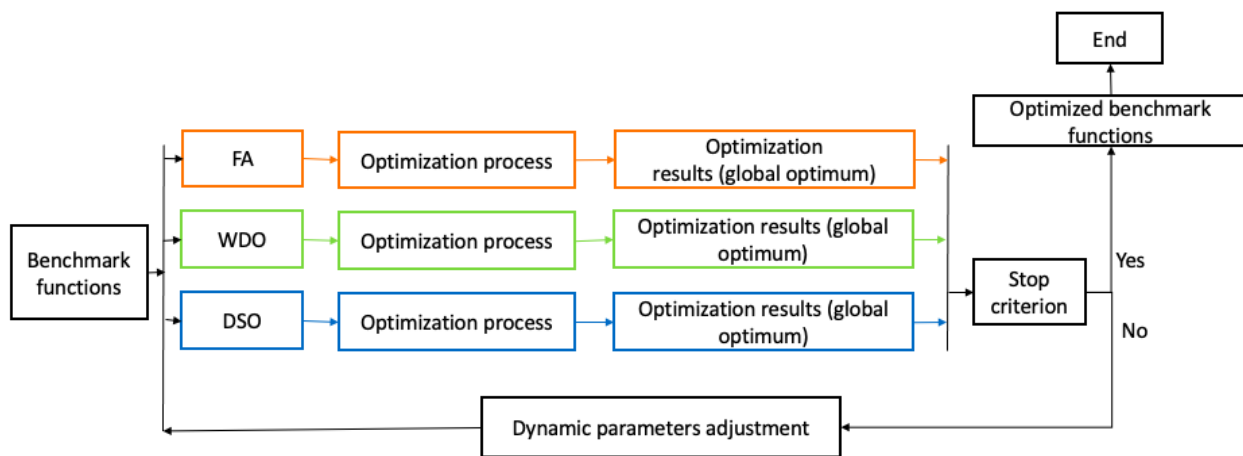


Fig. 4.23 Data flow of the optimization

4.2.1.1 Representation of the Benchmark functions to optimize.

The Unimodal, multimodal and Fixed-dimension multimodal benchmark functions were optimized, and their definitions are summarized in Tables 4.32, 4.33 and 4.34 respectively. The functions are optimized using the aforementioned methods and the competitiveness among them is compared, in this way finding out more in detail their operation and discovering advantages and disadvantages of each method.

Table 4.22 *Unimodal benchmark functions*

Benchmark functions $f_1 - f_7$

$$f_1(x) = \sum_{i=1}^n x_i^2$$

Search space $x_j \in [-100, 100]$ and $f(x^*) = 0$

$$f_2(x) = \sum_{i=1}^n |x_i| + \prod_{i=1}^n |x_i|$$

Search space $x_j \in [-10, 10]$ and $f(x^*) = 0$

$$f_3(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$$

Search space $x_j \in [-100, 100]$ and $f(x^*) = 0$

$$f_4(x) = \max_i \{|x_i|, 1 \leq i \leq n\}$$

Search space $x_j \in [-100, 100]$ and $f(x^*) = 0$

$$f_5(x) = \sum_{i=1}^{n-1} \left[100 (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$$

Search space $x_j \in [-30, 30]$ and $f(x^*) = 0$

$$f_6(x) = \sum_{i=1}^n ([x_i + 0.5])^2$$

Search space $x_j \in [-100, 100]$ and $f(x^*) = 0$

$$f_7(x) = \sum_{i=1}^n i x_i^4 + \text{random}[0,1]$$

Search space $x_j \in [-1.28, 1.28]$ and $f(x^*) = 0$

Table 4.23 Multimodal benchmark functions

Benchmark functions $f_9 - f_{11}$

$$f_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$$

Search space $x_j \in [-5.12, 5.12]$ and $f(x^*) = 0$

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

Search space $x_j \in [-32, 32]$ and $f(x^*) = 0$

$$f_{11}(x) = \frac{1}{400} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Search space $x_j \in [-600, 600]$ and $f(x^*) = 0$

Table 4.24 Fixed-dimension multimodal benchmark functions

Benchmark functions $f_{15} - f_{18}$

$$f_{15}(x) = \sum_{i=1}^{11} \left[a_i \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$$

Search space $x_j \in [-5, 5]$ and $f(x^*) = 0.00030$

$$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$$

Search space $x_j \in [-5, 5]$ and $f(x^*) = -1.0316$

$$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$$

Search space $x_j \in [-5, 5]$ and $f(x^*) = 0.398$

$$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ * [30 + (2x_1 - 3x_2)^2] * (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \\ \text{Search space } x_j \in [-2, 2] \text{ and } f(x^*) = 3$$

4.2.2 Results obtained from the optimization of benchmark functions

Table 4.35 shows the parameters used for the benchmark functions, column 1 of Table 4.36 shows the number that represents the particular function, column 2 shows the minimum of the function, columns 3 and 4 show the average and standard deviation respectively obtained with the WDO and finally in columns 5 and 6 the results of DSO are observed. Each of these functions were evaluated 30 times with the same parameters to obtain the averages and standard deviations.

Table 4.25 Parameters used in the experiments

Population	Iterations	Dimensions
30	500	30

Table 4.26 Results for 30 dimensions

Function	f _{mn}	WDO		DSO		z
		Average	Standard deviation	Average	Standard deviation	
f1	0	3.66x10 ⁻⁰	3.87E-27	4.95E-09	1.39E-08	-1.95x10 ⁺⁰⁰
f2	0	3.26E-14	2.70E-14	8.94E-09	2.68E-08	-1.83x10 ⁺⁰⁰
f3	0	2.42E-20	2.02E-20	1.16E-03	3.77E-03	-1.69x10 ⁺⁰⁰
f4	0	5.90E-13	6.18E-13	3.00E-03	6.76E-03	-2.43x10 ⁺⁰⁰
f5	0	2.86x10 ⁺⁰¹	7.65E-02	9.82x10 ⁺⁰⁰	1.15x10 ⁺⁰¹	8.94x10 ⁺⁰⁰
f6	0	1.74E-02	4.52E-03	2.71E-03	2.14E-05	1.78x10 ⁺⁰¹

f_7	0	1.12E-02	3.09E-02	2.71E-03	1.64E-03	$1.50x10^{+00}$
f_9	0	-118.27	0	$5.58x10^{+00}$	9.24E+00	$-7.34x10^{+01}$
f_{10}	0	9.53E-15	1.23E-14	3.47E-07	7.22E-07	$-2.63x10^{+00}$
f_{11}	0	0	0	3.66E-11	1.41E-10	$-1.42x10^{+00}$
f_{15}	0.0003	3.07E-04	2.18E-07	3.07E-04	1.39E-15	$0.00x10^{+00}$
f_{16}	-1.0316	$-1.03x10^{+00}$	6.78E-16	-1.03E+00	4.59E-16	$0.00x10^{+00}$
f_{17}	0.398	$3.98x10^{-01}$	6.78E-05	$3.98x10^{-01}$	0	0
f_{18}	3	$7.78x10^{+00}$	3.61E-15	3	3.26E-15	5.38E+15

The WDO average and standard deviation results with 64 dimensions can be compared with the results obtained with 30 dimensions.

Table 4.27 Parameters used in WDO, DSO and FA the methods

Population	Iterations	Dimensions
30	500	64

Table 4.28 Results for 64 dimensions

Function	f_{min}	WDO		DSO		z
		Average	Standard deviation	Average	Standard deviation	
f_1	0	$5.07x10^{-27}$	$8.47x10^{-27}$	$8.63x10^{-06}$	$2.16x10^{-05}$	$-2.19x10^{+00}$
f_2	0	$4.57x10^{-14}$	$5.89x10^{-14}$	$1.50x10^{-03}$	$3.99x10^{-03}$	$-2.06x10^{+00}$
f_3	0	$5.25x10^{-18}$	$5.66x10^{-18}$	$7.43x10^{-02}$	$2.04x10^{-01}$	$-1.99x10^{+00}$
f_4	0	$2.32x10^{-12}$	$1.68x10^{-12}$	$7.80x10^{-03}$	$1.35x10^{-02}$	$-5.48x10^{+00}$
f_5	0	$6.45x10^{-13}$	$5.42x10^{-13}$	$1.54x10^{+01}$	$2.34x10^{+01}$	$-5.48x10^{+00}$
f_6	0	$7.34x10^{-02}$	$1.36x10^{-02}$	$1.59x10^{-05}$	$2.41x10^{-05}$	$2.96x10^{+01}$
f_7	0	$5.11x10^{-02}$	$9.11x10^{-03}$	$5.79x10^{-03}$	$3.78x10^{-05}$	$2.30x10^{+01}$
f_9	0	$2.13x10^{+02}$	$4.66x10^{+01}$	$5.58x10^{+01}$	$5.22x10^{+01}$	$1.18x10^{+01}$
f_{10}	0	$3.85x10^{-15}$	$6.05x10^{-15}$	$1.30x10^{-03}$	$2.41x10^{-03}$	$-5.48x10^{+00}$
f_{11}	0	$2.12x10^{+01}$	$2.70x10^{-01}$	$2.96x10^{-07}$	$1.05x10^{-06}$	$4.30x10^{+02}$
f_{15}	0.0003	$3.08x10^{-04}$	$1.68x10^{-07}$	$3.07x10^{+00}$	$2.37x10^{-16}$	$1.78x10^{-02}$
f_{16}	-1.0316	$-1.03x10^{+00}$	$6.78x10^{-16}$	$-1.03x10^{+00}$	$4.50x10^{-16}$	$0.00x10^{+00}$
f_{17}	0.398	$3.98x10^{-01}$	$4.90x10^{-06}$	$3.98x10^{-01}$	$0.00x10^{+00}$	$0.00x10^{+00}$

f_{18}	3	$7.78x10^{+00}$	$3.61x10^{-15}$	3	$6.54x10^{-16}$	$8.73x10^{+00}$
----------	---	-----------------	-----------------	---	-----------------	-----------------

Table 4.37 above, it shows the optimization results of benchmark functions where the minimum of functions f_{15} , f_{16} , f_{17} and f_{18} is different from zero. With the evaluation of functions with different minima, the performance of the optimization in the functions with the WDO and DSO can be better observed.

Table 4.39 shows the parameters used in the methods mentioned above. These parameters are the same for each of these algorithms as a fair form of competition.

Table 4.29 Population, Iterations and Dimensions

Population	Iterations	Dimensions
30	500	128

Table 4.40 shows the results of 30 experiments, where the performance of the WDO and DSO method is compared, using 128 dimensions for the search evaluation of the objective function, in this case the minimum of each function that is in column 1 of the Table.

Table 4.30 Results obtained with 128 dimensions

Function	f_{min}	WDO		DSO		z
		Average	Standard deviation	Average	Standard deviation	
f_1	0	$4.35x10^{-27}$	$7.16x10^{-27}$	$1.49x10^{-04}$	$7.68x10^{-04}$	$-1.06x10^{+00}$
f_2	0	$3.88x10^{-14}$	$4.38x10^{-14}$	$2.66x10^{-02}$	$4.83x10^{-02}$	$-3.02x10^{+00}$
f_3	0	$1.09x10^{-16}$	$1.07x10^{-16}$	$1.21x10^{+02}$	$3.34x10^{+02}$	$-1.98x10^{+00}$
f_4	0	$1.52x10^{-12}$	$1.62x10^{-12}$	$2.56x10^{-02}$	$5.11x10^{-02}$	$-2.74x10^{+00}$
f_5	0	$7.68x10^{-13}$	$4.61x10^{-13}$	$4.60x10^{+01}$	$5.62x10^{+01}$	$-4.48x10^{+00}$
f_6	0	$1.85x10^{-01}$	$3.26x10^{-02}$	$7.43x10^{-01}$	$7.34x10^{-01}$	$-4.09x10^{+00}$
f_7	0	$1.66x10^{-01}$	$3.11x10^{-02}$	$6.83x10^{-03}$	$6.49x10^{-03}$	$2.74x10^{+01}$
f_9	0	$4.58Ex10^{+02}$	$2.09x10^{+02}$	$8.38x10^{+01}$	$1.19x10^{+02}$	$8.52x10^{+00}$
f_{10}	0	$8.88x10^{-16}$	$4.00x10^{-16}$	$2.12x10^{-04}$	$7.98x10^{-04}$	$-1.46x10^{+00}$
f_{11}	0	$2.07x10^{+01}$	$3.91x10^{+00}$	$1.48x10^{-03}$	$3.93x10^{-03}$	$2.90x10^{+01}$
f_{15}	0.0003	$3.08x10^{-04}$	$1.93x10^{-07}$	$3.07x10^{-04}$	$1.39x10^{-16}$	$2.84x10^{+01}$
f_{16}	-1.0316	$-1.03x10^{+00}$	$8.78x10^{-16}$	$-1.03x10^{+00}$	$4.70x10^{-16}$	$0.00x10^{+00}$
f_{17}	0.398	$3.98x10^{-01}$	$3.46x10^{-06}$	$3.98x10^{-01}$	$0.00x10^{+00}$	$0.00x10^{+00}$
f_{18}	3	$7.78x10^{+00}$	$3.61x10^{-15}$	$3.00x10^{+00}$	$4.79x10^{-06}$	$5.47x10^{+06}$

The following Tables 4.41 and 4.42 show the results obtained with the FA for the optimization of the f1 and f2 function with 30, 64 and 128 dimensions, where it can be noted that the results are very far from the global minima of the functions. Previously we have been experimenting with this FA method where we can note that to obtain a good result with it, it is necessary to increase the iterations and maintain a population of 30 to 50 fireflies. Therefore, as future work in this methodology an adjustment will be made to the method to improve its behavior in the optimization of benchmark functions, remembering that this methodology is proposed just for that, to find which method is good for a specific problem and in competitiveness with others, and the method with which worse result you get to make an improvement to help your behavior.

Table 4.31 Results for f1 with FA

Function	f1	f1	f1
Iterations	500	500	500
Dimensions	30	64	128
Average	$8.80x10^{-03}$	$8.40x10^{-03}$	$3.67x10^{-01}$
Standar Desviation	$1.90x10^{-03}$	$1.20x10^{-02}$	$2.93x10^{-02}$

Table 4.32 Results for f2 with FA

Function	f2	f2	f2
Iterations	500	500	500
Dimensions	30	64	128
Average	$3.20x10^{-01}$	$1.84x10^{+00}$	$1.45x10^{+00}$
Standar Desviation	$6.92x10^{-02}$	$2.58x10^{-01}$	$2.00x10^{-01}$

4.3 Optimization of the autonomous robot

For this third case, the proposed model for optimization is the one described above in case 4.2.1 where 3 different methods are used to find the optimal data vector for optimizing the membership functions of the fuzzy controller.

4.3.1 Optimization of a mobile autonomous robot tracking line

Optimization for a fuzzy controller of an autonomous mobile robot, using the multi-metaheuristic competitiveness model to find out which is the satisfactory optimization algorithm for this specific problem, and below the fuzzy controller is explained in [50]

This autonomous robot mobile controller in [50], and [51] also describes applications of the method, aims to accurately track a given desired trajectory, has 9 rules that allow to create a relationship between the linguistic variables of the fuzzy system. The linguistic variables in the first input is the linear velocity (ev), in the second and last input is the angular velocity (ew), the two, namely the inputs have the same linguistic values: Positive (P), Zero (Z) and negative (N), with the same membership functions in each one. The outputs are two Torque 1 ($t1$) and Torque 2 ($t2$) that represent the movement of the wheels when the robot rotates, and each of these outputs has 3 triangular functions.

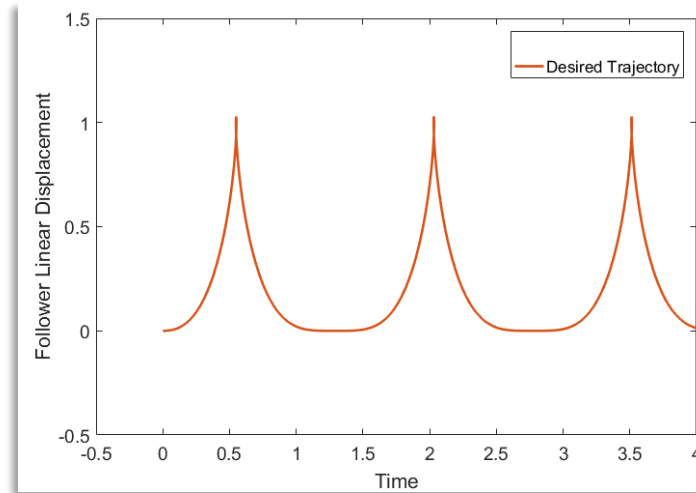


Fig. 4.24 Desired trajectory, normalized axis.

Figure 4.36 shows the desired trajectory that the robot has to follow, this figure has the Y axis of the desired displacement line, and the X axis shows the displacement time.

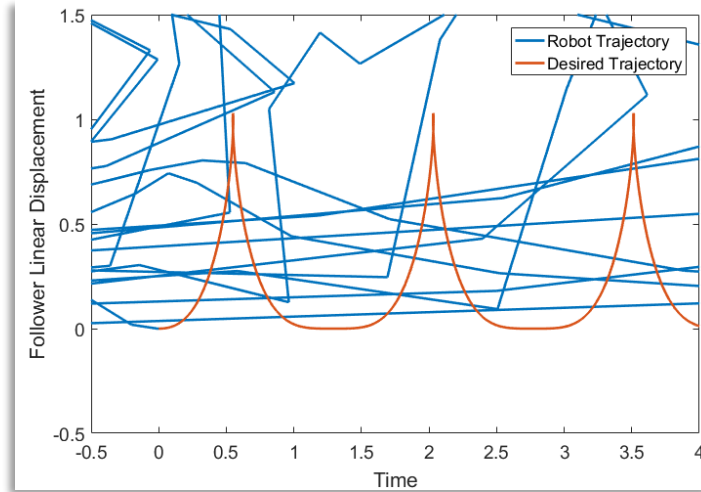


Fig. 4.25 Obtained MSE error

Figure 4.37 shows the red line as the desired trajectory, and in blue the actual trajectory of the robot, as can be seen the robot is very lost since it generates an error of $3.8541 \times 10^{+03}$.

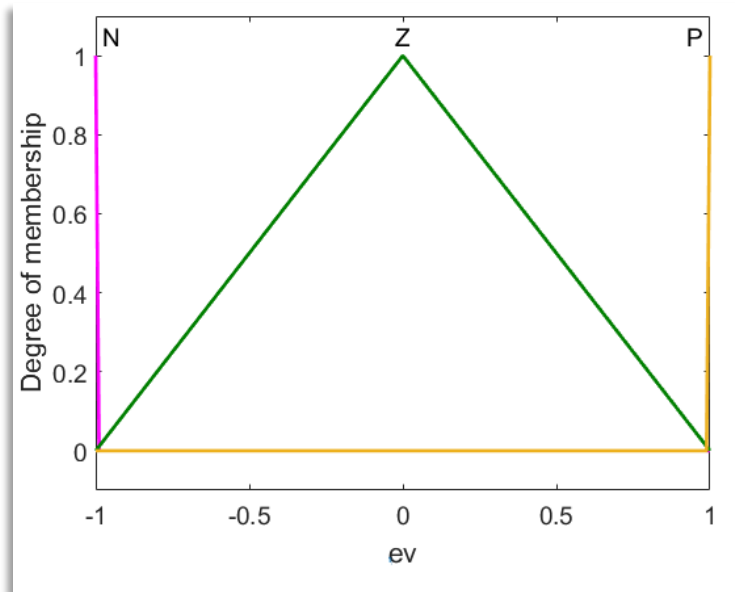


Fig. 4.26. Input ev, not optimized

In Figure 4.38, Illustrated the membership functions of input 1, which is called linear speed, where its linguistic variables represent negative, zero and positive.

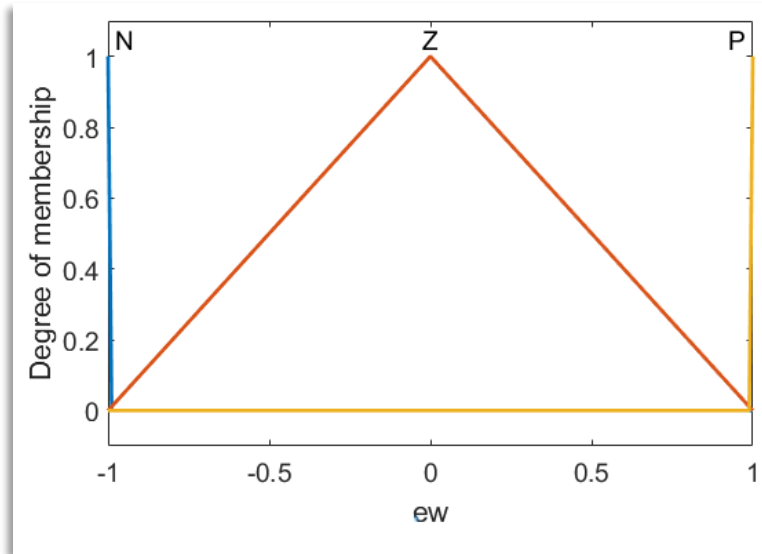


Fig. 4.27 Input ew, not optimized

Figure 4.39, you can see the membership functions, the blue color corresponds to a trapezoidal, the orange represents the zero linguistic variable and is a triangular membership function, the yellow line is a trapezoidal membership function of input 2 of the fuzzy controller.

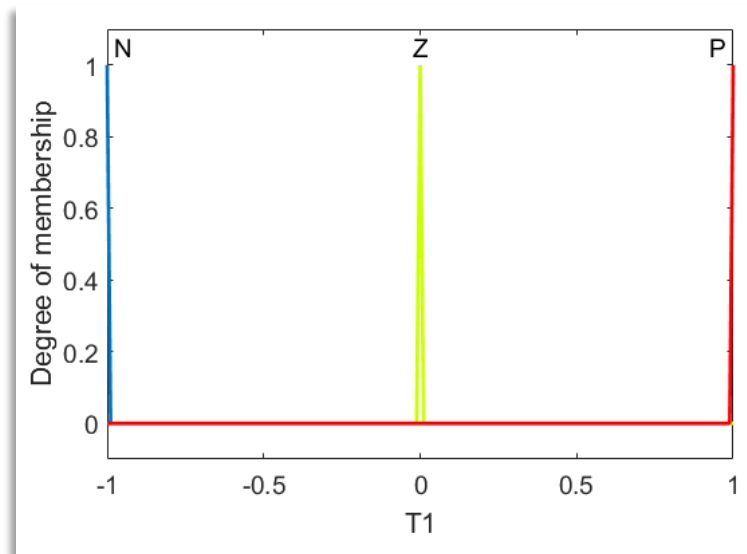


Fig. 4.28 Output T1, not optimized

Figure 4.40 shows the 3 triangular membership functions of the output 1 represents the movement of the wheel 1 of the fuzzy control, as illustrated by the functions of not observed well defined, because the controller is not optimized in this figure.

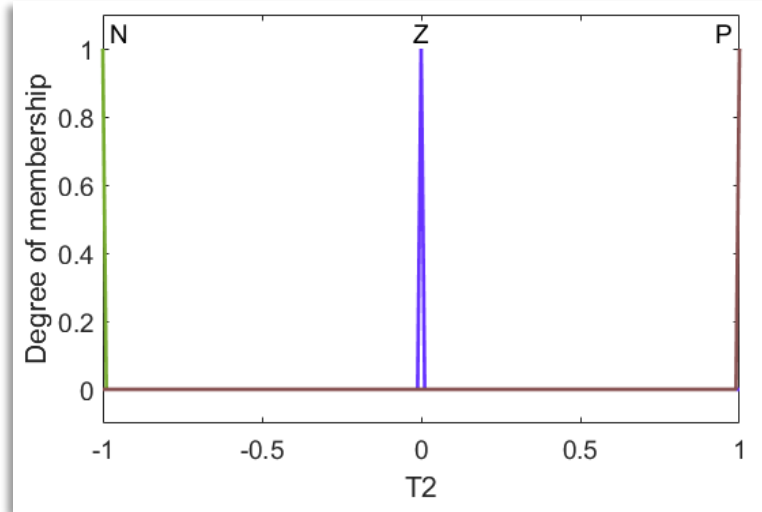


Fig. 4.29 Output T2, not optimized

Figure 4.41 represents the output 2 of the fuzzy controller (T2) is the movement of the wheel 1, has as linguistic variables negative, zero and positive in this case the three membership functions are triangular.

Table 4.33 fuzzy if then rules

Fuzzy if then rules
1. If (ev is N) and (ew is N) then (T1 is N)(T2 is N) (1)
2. If (ev is N) and (ew is Z) then (T1 is N)(T2 is Z) (1)
3. If (ev is N) and (ew is P) then (T1 is N)(T2 is P) (1)
4. If (ev is Z) and (ew is N) then (T1 is Z)(T2 is N) (1)
5. If (ev is Z) and (ew is Z) then (T1 is Z)(T2 is Z) (1)
6. If (ev is Z) and (ew is P) then (T1 is Z)(T2 is P) (1)
7. If (ev is P) and (ew is N) then (T1 is P)(T2 is N) (1)
8. If (ev is P) and (ew is Z) then (T1 is P)(T2 is Z) (1)
9. If (ev is P) and (ew is P) then (T1 is P)(T2 is P) (1)

Table 4.43 shows the representation of the if-then rules of the fuzzy controller that describe the relationship between the linguistic variables. Where ev represents the linear velocity and ew is the angular velocity.

4.3.2 Results of the optimization made to the robot autonomous

This section presents the results of the optimization of the robot autonomous using multi-metaheuristics model.

4.3.2.1 Fuzzy controller optimization results

Table 4.34 Parameters used in FA, WDO and DSO for fuzzy controller optimization

Population	Iterations
20	1500

Table 4.44 shows the parameters used in the FA, WDO, DSO methods and Table 4.45 shows 30 experiments performed to obtain the best optimized fuzzy system, using as metric the MSE, where it can be observed that the best error found is of 0.00169, in general the values only varied from 10^{-02} to 10^{-03} .

Table 4.35 Results of the experiments with the fuzzy controller

Experiment	MSE	Experiment	MSE
1	1.96×10^{-03}	16	2.99×10^{-02}
2	4.80×10^{-03}	17	3.33×10^{-02}
3	6.31×10^{-03}	18	4.09×10^{-02}
4	6.33×10^{-03}	19	4.32×10^{-02}
5	8.25×10^{-03}	20	5.33×10^{-02}
6	9.09×10^{-03}	21	7.11×10^{-02}
7	9.58×10^{-03}	22	7.06×10^{-02}
8	1.71×10^{-02}	23	7.72×10^{-02}
9	1.82×10^{-02}	24	8.03×10^{-02}
10	1.85×10^{-02}	25	8.47×10^{-02}
11	1.98×10^{-02}	26	8.75×10^{-02}
12	2.11×10^{-02}	27	8.97×10^{-02}
13	2.27×10^{-02}	28	9.16×10^{-01}
14	2.45×10^{-02}	29	9.42×10^{-02}
15	2.52×10^{-02}	30	9.63×10^{-02}

In Figures 4.42 and 4.43, the variation that was made in the parameters of the membership functions of the two inputs of the fuzzy controller is observed, where the uncertainty that exists between each of them can be observed, with this considerably improving the error that generates the fuzzy controller in the simulation. Figures 4.44 and 4.45 show the outputs of the controller optimized by the FA, respectively.

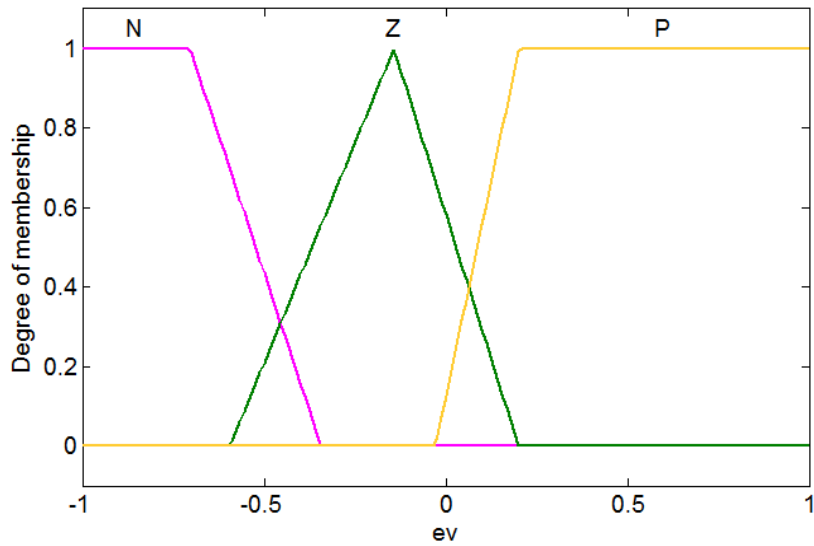


Fig. 4.30 Input 1 (ev)

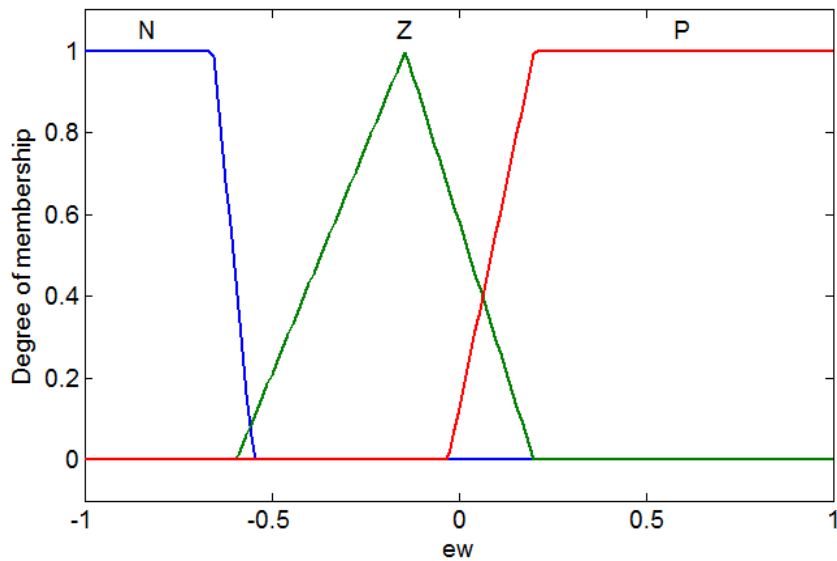


Fig. 4.31 Input 2 (*ew*)

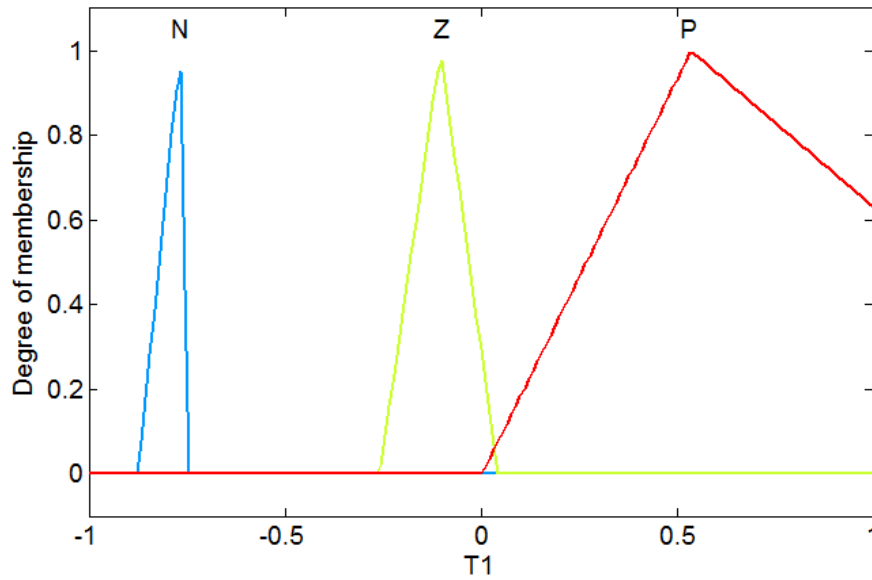


Fig. 4.32 Output 1 (T1)

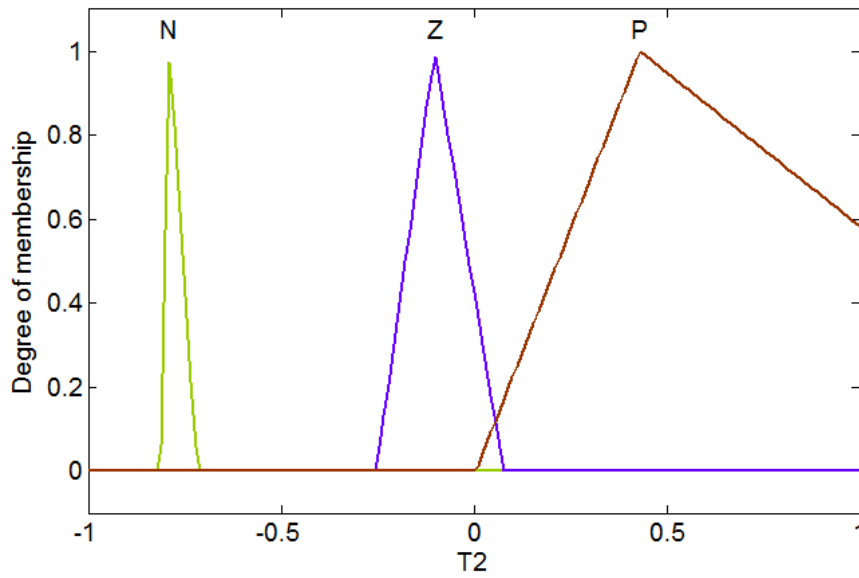


Fig. 4.33 Output 2 (T2)

As can be noted in Table 4.46, the results obtained with the WDO method were very far from each other, since it has the best MSE of 0.000019, but as the worst is 0.057094, and for this reason the average method cannot be the best at optimizing this specific problem.

Table 4.36 Results obtained with WDO

Experiment	MSE	Experiment	MSE
------------	-----	------------	-----

1	1.90×10^{-05}	16	1.64×10^{-04}
2	1.90×10^{-05}	17	1.64×10^{-04}
3	5.40×10^{-05}	18	2.60×10^{-04}
4	5.40×10^{-05}	19	2.93×10^{-04}
5	6.20×10^{-05}	20	4.33×10^{-04}
6	6.20×10^{-05}	21	1.62×10^{-03}
7	6.90×10^{-05}	22	1.65×10^{-03}
8	8.70×10^{-05}	23	6.21×10^{-03}
9	8.70×10^{-05}	24	6.35×10^{-03}
10	9.10×10^{-05}	25	7.73×10^{-03}
11	9.80×10^{-05}	26	1.67×10^{-02}
12	1.05×10^{-04}	27	1.67×10^{-02}
13	1.15×10^{-04}	28	2.65×10^{-02}
14	1.38×10^{-04}	29	2.92×10^{-02}
15	1.60×10^{-04}	30	5.70×10^{-02}

Figures 4.46, 4.47, 4.48 and 4.49 show the optimized movement of the parameters of the membership functions of the inputs ev , ew and the outputs $t1$ and $t2$, respectively.

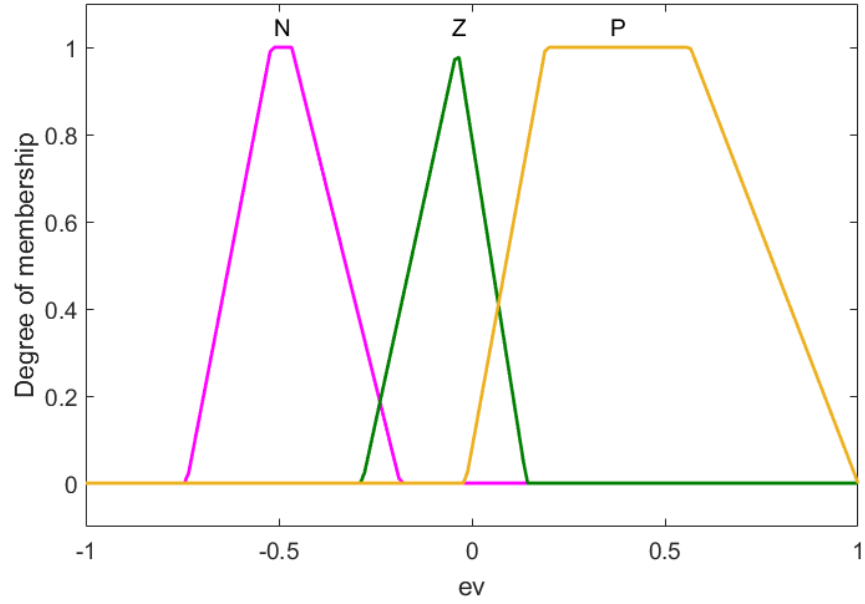


Fig. 4.34 Membership functions for Input 1

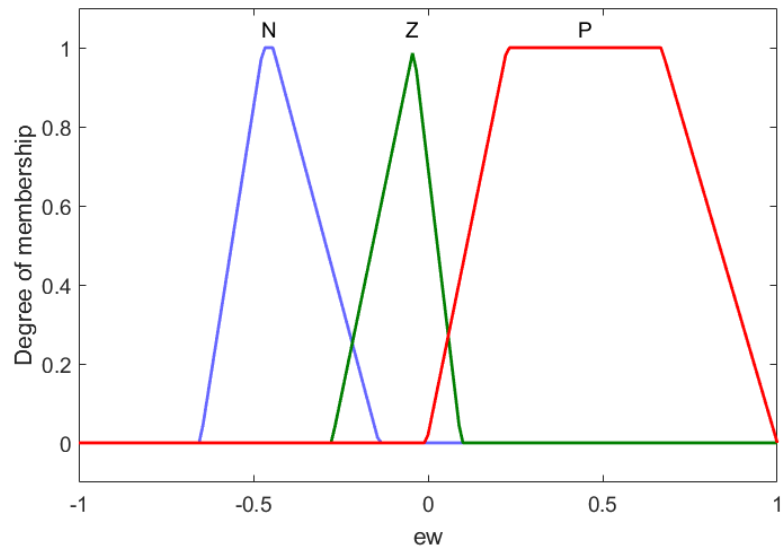


Fig. 4.35 Membership functions for Input 2

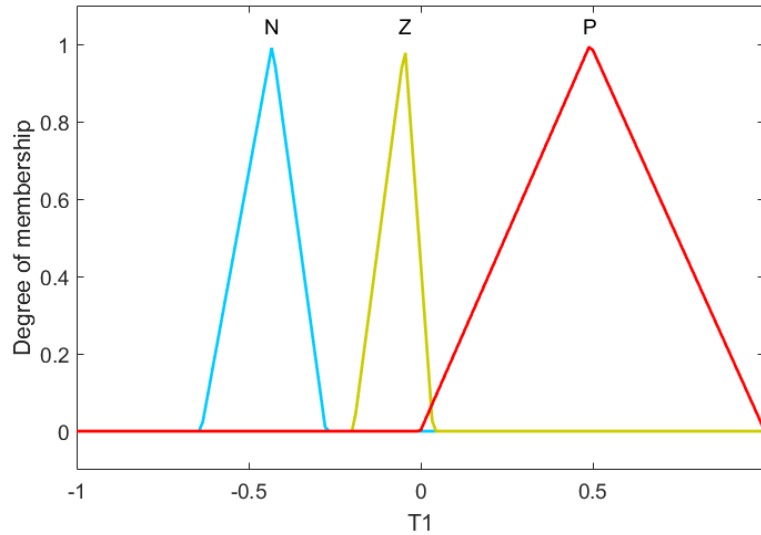


Fig. 4.36 Membership functions for Output 1

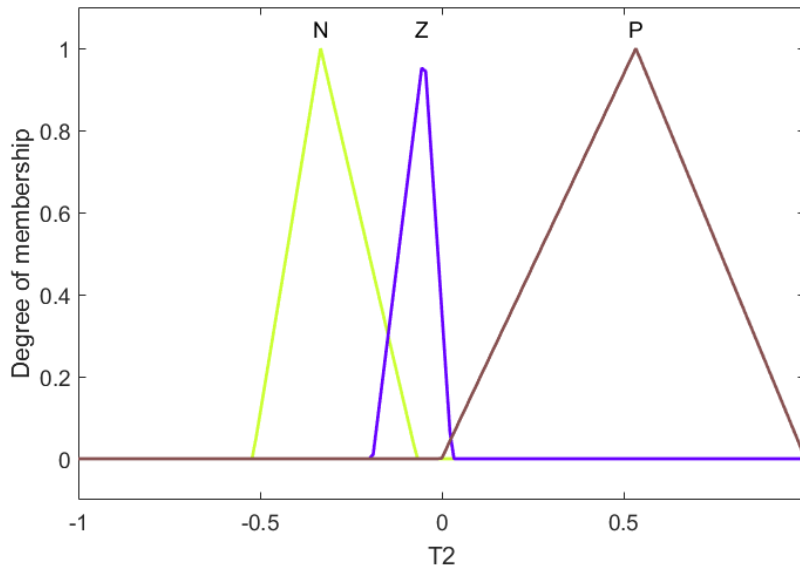


Fig. 4.37 Membership functions for Output 2

The DSO on average gives a better result than the FA and WDO as can be noted in Table 4.47 column 2, it appears that this is not the case, since the WDO method gives much lower values, but they are very separated from each other and the DSO remain consistent with the delivered results of the MSE.

Table 4.37 Results obtained with DSO

Experiment	MSE	Experiment	MSE
1	$1.69x10^{-03}$	16	$2.99x10^{-02}$
2	$4.80x10^{-03}$	17	$3.33x10^{-02}$
3	$6.31x10^{-03}$	18	$4.09x10^{-02}$
4	$6.33x10^{-03}$	19	$4.32x10^{-02}$
5	$8.25x10^{-03}$	20	$5.33x10^{-02}$
6	$9.09x10^{-03}$	21	$7.11x10^{-02}$
7	$9.58x10^{-03}$	22	$7.60x10^{-02}$
8	$1.71x10^{-02}$	23	$7.72x10^{-02}$
9	$1.82x10^{-02}$	24	$8.03x10^{-02}$
10	$1.85x10^{-02}$	25	$8.04x10^{-02}$
11	$1.98x10^{-02}$	26	$8.75x10^{-02}$
12	$2.11x10^{-02}$	27	$8.97x10^{-02}$
13	$2.27x10^{-02}$	28	$9.16x10^{-02}$
14	$2.45x10^{-02}$	29	$9.42x10^{-02}$
15	$2.52x10^{-02}$	30	$9.63x10^{-02}$

As can be noted from Figures 4.50, 4.51, 4.52 and 4.53 the overlap between existing functions helps the robot to have a better tracking of the desired trajectory, in comparison with the other methods used in this methodology.

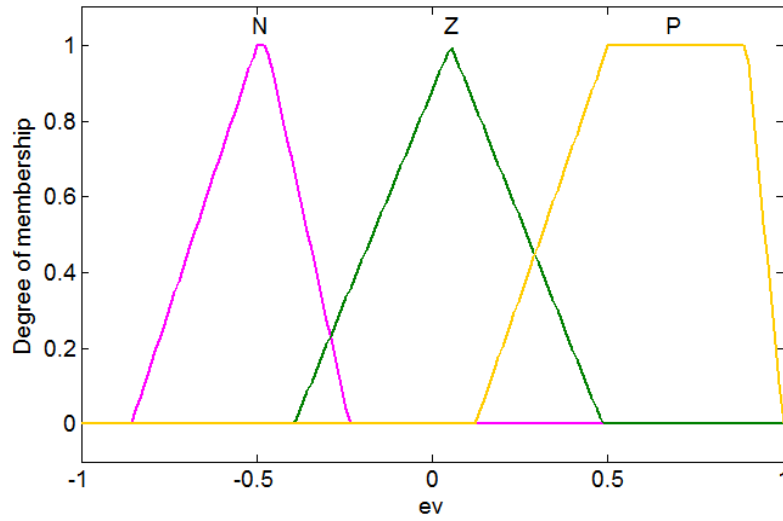


Fig. 4.38 Input 1 optimized by DSO

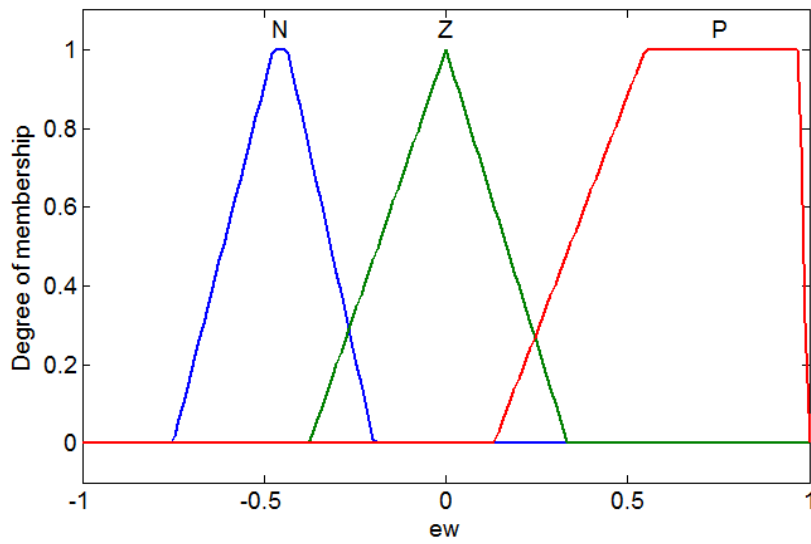


Fig. 4.39 Input 2 optimized by DSO

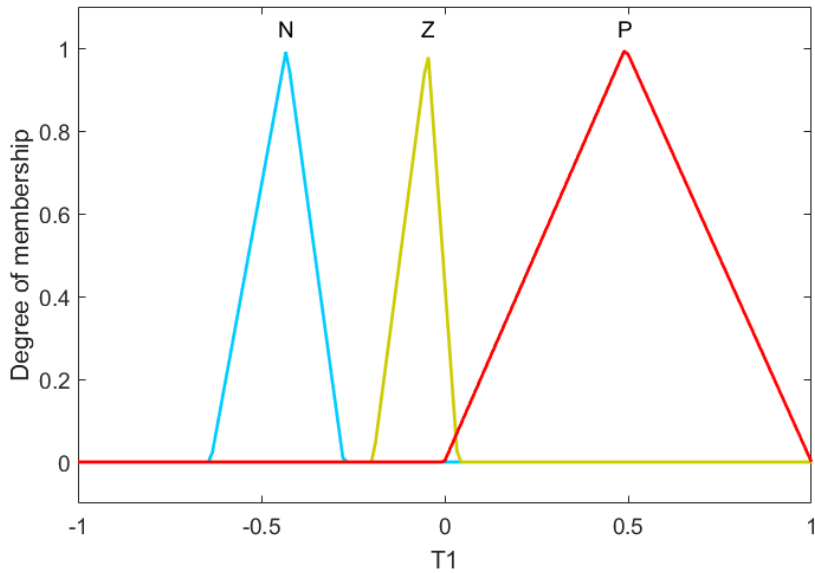


Fig. 4.40 Output 1 optimized by DSO

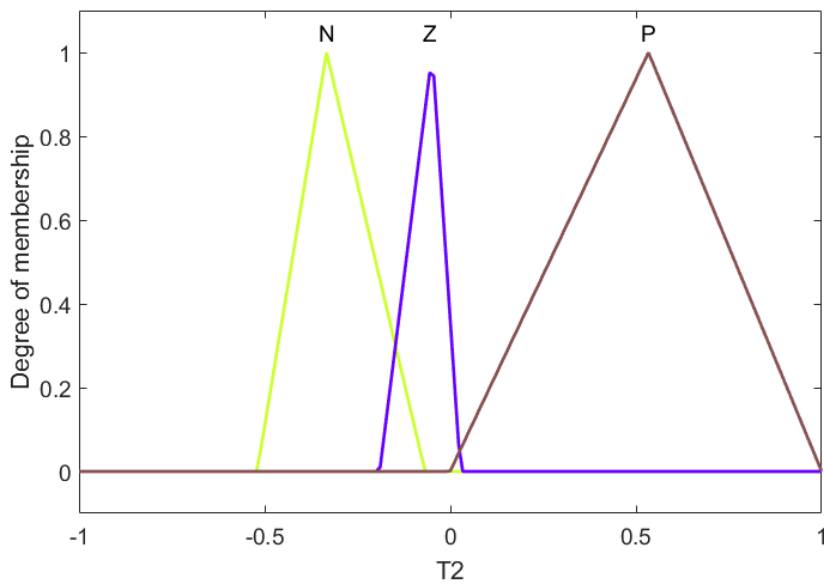


Fig. 4.41 Output 1 optimized by DSO

4.3.2.2 Comparative analysis and hypothesis testing of optimization results

Table 4.38 Summary of results from the optimization methods

	FA	WDO	DSO

Average	4.21×10^{-02}	5.78×10^{-03}	5.02×10^{-02}
Standard deviation	3.31×10^{-02}	1.24×10^{-02}	3.3×10^{-02}

The equation for Z test is:

$$Z = \frac{x_1 - x_2 - (1 - 2) \times 1 - x_2}{\dots} \quad (4.8)$$

Table 4.39 Mean and Deviation Standard

Variable	Number of samples	Mean	Desviation Standard
WDO	30	0.006	0.012
DSO	30	0.042	0.033

In the Table 4.50 shows the statistical data used in the z test

Table 4.40 Parameters statistical test

Difference	-0.036
z (Observed value)	-5.628
z (Critica value)	-1.645
valor-p (one-tailed)	< 0.0001
alpha	0.05

The methodology proposes competitiveness among optimization metaheuristics to improve overall performance, and this is aimed at reducing the search time from one method to another when one of them does not give the expected results.

In addition to the search time, it is also obtained with certainty which method is better, for which type of problem optimization, since the competitiveness that exists between them delivers only the best result. With this methodology, the method that generates the worst result is also

helped, also contributing to other scientists to observe in which cases the used methods are good and in which are not. Therefore, with the obtained results in the optimization of unimodal, multimodal and Fixed-dimension multimodal benchmark functions, it can be said that the WDO in general gives, on average, better results for this type of functions, while the DSO algorithm gives more results far from the minimum of the functions. However, in terms of parameter optimization of the membership functions of a fuzzy controller in particular, as can be noted in Table 18, the method that on average produces better results is DSO.

The method that can be said to be the worst in these results of competitiveness is the FA, since this method did not show good results with any of the two previous cases of minimization. For this reason, as future work it is proposed to make an improvement to this method, depending on the previous research that has its disadvantages in these cases.

4.4 A new Approach for Dynamic Stochastic Fractal Search with Fuzzy Logic for Parameter Adaptation

For this fourth case study, the adaptation of the parameters for the SFS method is carried out, using the following methodology [52]:

4.5 Optimization to Dynamic Stochastic Fractal Search with Fuzzy Logic for Parameter Adaptation

As mentioned above the SFS [53] has two important processes: the diffusion method and the updating strategy. Analyzing the diffusion method we know that a particle called seed originates, and others are generated and adhered to it, through diffusion that forms a chaotic-stochastic fractal branch, taking into account this fact the stochastic fractal search (SFS) method was improved by adding diversity to the diffusion process for the particles in each iteration, in this way, the Gaussian random walk is helped, where the particles have more possibilities to exploit the search space and therefore do not stagnate in an optimal location. To control the diffusion parameter, a fuzzy inference system was introduced which dynamically adjusts the diffusion of the particles with a range of [0 1], this fuzzy system for control has two inputs (iteration and diversity) and one output (diffusion) as illustrated in Figure 4.54. The fuzzy system implemented

has 9 if-then rules which allow to express the knowledge available about the relationship between antecedents and consequents. These rules determine the behavior of the fuzzy controller and it is here output parameter is emulated, the idea is to have an efficient algorithm that improves the disadvantages of the SFS, which is capable of producing solutions of good quality. In this case, Equations. (11) and (12). Represent diversity and iteration, respectively.

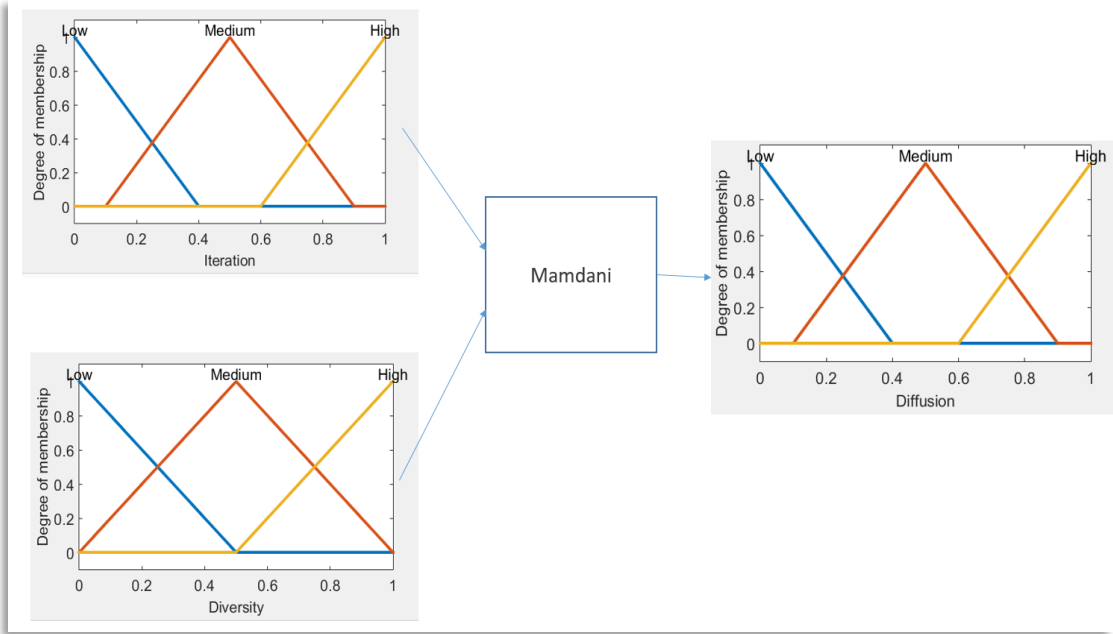


Fig. 4.42 Type-1 fuzzy system for controlling of diffusion

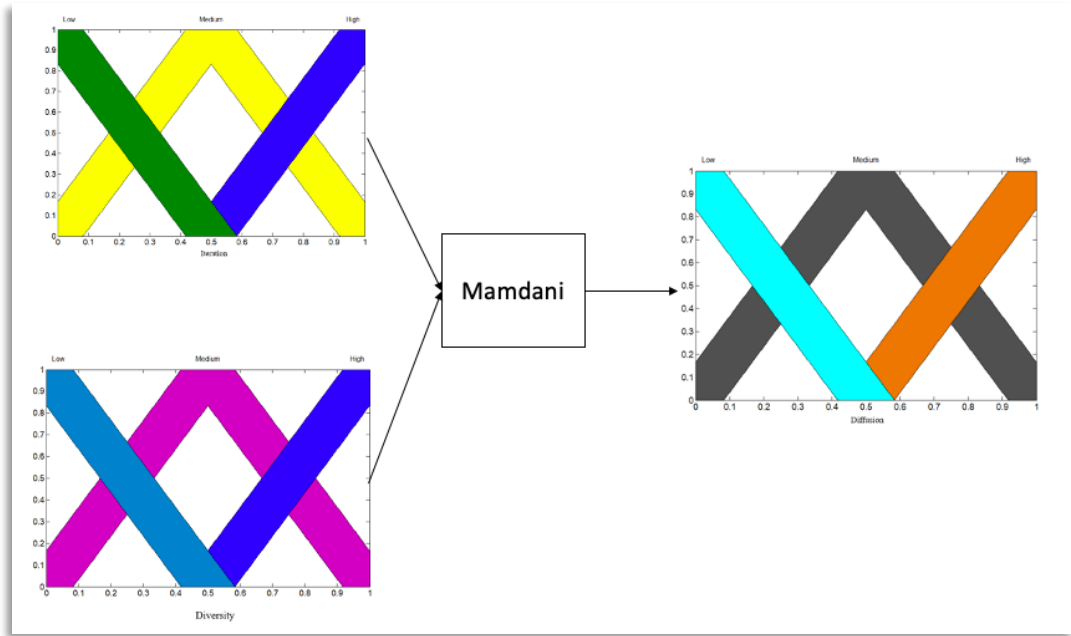


Fig. 4.43 Type-1 fuzzy system for controlling of diffusion

Fuzzy if then rules

If (Iteration is Low) and (Diversity is Low) then (Diffusion is High)

If (Iteration is Low) and (Diversity is Medium) then (Diffusion is Medium)

If (Iteration is Low) and (Diversity is High) then (Diffusion is Medium)

If (Iteration is Medium) and (Diversity is Low) then (Diffusion is Medium)

If (Iteration is Medium) and (Diversity is Medium) then (Diffusion is Medium)

If (Iteration is Medium) and (Diversity is High) then (Diffusion is Medium)

If (Iteration is High) and (Diversity is Low) then (Diffusion is Medium)

If (Iteration is High) and (Diversity is Medium) then (Diffusion is Medium)

If (Iteration is High) and (Diversity is High) then (Diffusion is Low)

Equations. 9 and 10. Mathematically define the geometrical shape of the triangular functions, for type-1 and type-2 fuzzy logic, respectively [54], [55].

$$triangular(u; a, b, c) = \begin{cases} 0, & u \leq a \\ \frac{u-a}{b-c}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & c \leq x \end{cases} \quad (4.9)$$

$$\begin{aligned} \tilde{\mu}(x) &= [\underline{\mu}(x), \bar{\mu}(x)] = \text{itrapatype2}(x, [a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, \alpha]) \\ &\text{where } a_1 < a_2, b_1 < b_2, c_1 < c_2, d_1 < d_2 \\ \mu_1(x) &= \max\left(\min\left(\frac{x-a_1}{b_1-a_1}, 1, \frac{d_1-x}{d_1-c_1}\right), 0\right) \\ \mu_2(x) &= \max\left(\min\left(\frac{x-a_2}{b_2-a_2}, 1, \frac{d_2-x}{d_2-c_2}\right), 0\right) \\ \bar{\mu}(x) &= \begin{cases} \max(\mu_1(x), \mu_2(x)) & \forall x \notin (b_1, c_2) \\ 1 & \forall x \in (b_1, c_2) \end{cases} \\ \underline{\mu}(x) &= \min\left(\alpha, \min(\mu_1(x), \mu_2(x))\right) \end{aligned} \quad (4.10)$$

$$Iteration = \frac{Current\ Iteration}{Total\ of\ Iterations} \quad (4.11)$$

$$Diversity(S(t)) = \frac{1}{n} \sum_{i=1}^n \sqrt{\sum_{j=1}^D [x_{ij}(t) - \bar{x}_j(t)]^2} \quad (4.12)$$

Diversity contributes to the stochastic movement of the particles, to having more possibility of exploiting the entire search space, in each iteration it is dynamically adjusted close to the global optimum, thus the diffusion process is improved and therefore the most efficient method.

4.5.1 Results obtained from the optimization

To obtain an understanding of the effectiveness of the dynamic stochastic fractal search (DSFS) method, 30 functions of the CEC'2017 competition [56], summarized in Table 4.59, were evaluated. In Table 4.59 we can find several types of functions, such as: unimodal, multimodal, hybrid and composite. To compare the optimization performance of the proposal respective to other methods, different number of dimensions (10, 30, 50 and 100) and different adaptation strategies for the parameters (using type-1 and type-2 fuzzy systems) were used.

Table 4.41 CEC 2017 optimization functions

	No	Function	fi
Unimodal Functions	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Sum of Different Power Function	200
	3	Shifted and Rotated Zakharov Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Rastrigin's Function	500
	6	Shifted and Rotated Expanded Scaffer's Function	600
	7	Shifted and Rotated Lunacek Bi_Rastrigin Function	700
	8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	9	Shifted and Rotated Levy Function	900
	10	Shifted and Rotated Schwefel's Function	1000
Hybrid Functions	11	Hybrid Function 1 (N = 3)	1100
	12	Hybrid Function 2 (N = 3)	1200
	13	Hybrid Function 3 (N = 3)	1300
	14	Hybrid Function 4 (N = 4)	1400
	15	Hybrid Function 5 (N = 4)	1500
	16	Hybrid Function 6 (N = 4)	1600
	17	Hybrid Function 6 (N = 5)	1700
	18	Hybrid Function 6 (N = 5)	1800
	19	Hybrid Function 6 (N = 5)	1900
	20	Hybrid Function 6 (N = 6)	2000
	21	Composition Function 1 (N = 3)	2100
22	Composition Function 2 (N = 3)	2200	
23	Composition Function 3 (N = 4)	2300	
24	Composition Function 4 (N = 4)	2400	
25	Composition Function 5 (N = 5)	2500	

Composition Functions	26	Composition Function 6 (N = 3)	2600
	27	Composition Function 7 (N = 6)	2700
	28	Composition Function 8 (N = 3)	2800
	29	Composition Function 9 (N = 3)	2900
	30	Composition Function 10 (N = 3)	3000

To evaluate the proposed method, the CEC'2017 benchmark mathematical functions that have been widely used in the literature are considered in the tests. Table 4.60 shows at the top the dimensions used for each function, column 1 shows the function that is given as 1,2,3,4 ... 29., column 2 represented as f1 shows the global optimum (minimum) of each function, the next two columns contain the results obtained by the dynamic stochastic fractal search (DSFS) algorithm using type-1 fuzzy logic, the observable results are the average and standard deviation of each of the functions. Finally, columns 5 and 6 show the mean and standard deviations, respectively, of the functions when using type-2 fuzzy logic for the adaptation of the parameter values in DSFS.

Table 4.42 Dynamic Stochastic Fractal Search DSFS results with 10 dimensions

Dynamic Stochastic Fractal Search DSFS with 10 Dimensions					
Function	fi	Type 1 Fuzzy logic		Type 2 Fuzzy logic	
		Main	Std	Main	Std
f1	100	1.01 $x10^{+02}$	$4.87x10^{-01}$	1.01 $x10^{+02}$	$7.85x10^{-01}$
f2	200	2.00 $x10^{+02}$	$0.00x10^{+00}$	2.00 $x10^{+02}$	$0.00x10^{+00}$
f3	300	3.00 $x10^{+02}$	$3.73x10^{-06}$	3.00 $x10^{+02}$	$5.04x10^{-06}$
f4	400	4.01 $x10^{+02}$	$7.81x10^{-01}$	4.00 $x10^{+02}$	$6.19x10^{-01}$
f5	500	5.06 $x10^{+02}$	$2.04x10^{+00}$	5.07 $x10^{+02}$	$2.64x10^{+00}$
f6	600	6.00 $x10^{+02}$	$7.21x10^{-08}$	6.00 $x10^{+02}$	$6.24x10^{-08}$
f7	700	7.19 $x10^{+02}$	$3.45x10^{+00}$	7.20 $x10^{+02}$	$3.32x10^{+00}$
f8	800	8.07 $x10^{+02}$	$3.04x10^{+00}$	8.07 $x10^{+02}$	$2.44x10^{+00}$
f9	900	9.00 $x10^{+02}$	$0.00x10^{+00}$	9.00 $x10^{+02}$	$0.00x10^{+00}$
f10	1000	1.40 $x10^{+03}$	$1.70x10^{+02}$	1.34 $x10^{+03}$	$1.31x10^{+02}$
f11	1100	1.10 $x10^{+03}$	$9.31x10^{-01}$	1.10 $x10^{+03}$	$1.00x10^{+00}$
f12	1200	1.50 $x10^{+03}$	$1.04x10^{+02}$	1.52 $x10^{+03}$	$1.02x10^{+02}$
f13	1300	1.31 $x10^{+03}$	$4.03x10^{+00}$	1.31 $x10^{+03}$	$4.14x10^{+00}$
f14	1400	1.40 $x10^{+03}$	$1.69x10^{+00}$	1.40 $x10^{+03}$	$2.01x10^{+00}$
f15	1500	1.50 $x10^{+03}$	$7.24x10^{-01}$	1.50 $x10^{+03}$	$8.88x10^{-01}$
f16	1600	1.60E	$2.79x10^{-01}$	1.60 $x10^{+03}$	$3.64x10^{-01}$
f17	1700	1.70 $x10^{+03}$	$2.68x10^{+00}$	1.71 $x10^{+03}$	$4.41x10^{+00}$
f18	1800	1.81 $x10^{+03}$	$2.56x10^{+00}$	1.81 $x10^{+03}$	$2.57x10^{+00}$
f19	1900	1.90 $x10^{+03}$	$4.15x10^{-01}$	1.90 $x10^{+03}$	$4.45x10^{-01}$

f20	2000	2.00 $\times 10^{+03}$	1.39×10^{-01}	2.00 $\times 10^{+03}$	4.73×10^{-03}
f21	2100	2.23 $\times 10^{+03}$	$5.08 \times 10^{+01}$	2.24 $\times 10^{+03}$	$5.40 \times 10^{+01}$
f22	2200	2.29 $\times 10^{+03}$	$2.17 \times 10^{+01}$	2.28 $\times 10^{+03}$	$3.93 \times 10^{+01}$
f23	2300	2.61 $\times 10^{+03}$	$2.80 \times 10^{+00}$	2.60 $\times 10^{+03}$	$5.61 \times 10^{+01}$
f24	2400	2.61 $\times 10^{+03}$	$1.22 \times 10^{+02}$	2.63 $\times 10^{+03}$	$1.14 \times 10^{+02}$
f25	2500	2.90 $\times 10^{+03}$	$1.08 \times 10^{+01}$	2.90 $\times 10^{+03}$	$1.15 \times 10^{+01}$
f26	2600	2.90 $\times 10^{+03}$	2.15×10^{-01}	2.90 $\times 10^{+03}$	3.02×10^{-01}
f27	2700	3.09 $\times 10^{+03}$	$1.97 \times 10^{+00}$	3.09 $\times 10^{+03}$	$2.14 \times 10^{+00}$
f28	2800	3.09 $\times 10^{+03}$	$4.20 \times 10^{+01}$	3.11 $\times 10^{+03}$	$5.69 \times 10^{+01}$
f29	2900	3.16 $\times 10^{+03}$	$1.01 \times 10^{+01}$	3.16 $\times 10^{+03}$	$9.65 \times 10^{+00}$
f30	3000	3.56 $\times 10^{+03}$	$2.30 \times 10^{+02}$	3.55 $\times 10^{+03}$	$1.19 \times 10^{+02}$

As can be seen, the results using type 1 and type 2 fuzzy systems did not have a significant visible difference using 10 dimensions, even so the results obtained were good on average, because most reached the optimal global of the functions. Table 4.61, row 3 shows in the first column, the number of function that is being evaluated, in column two, the optimum of each function is observed, in column 3, the average results of each of the functions are given, the functions, the four shows the standard deviation, these two obtained with type 1 fuzzy logic, and the last columns illustrate the results in average and standard deviation respectively, with type 2 fuzzy logic, the difference between these results was minimum even though these results were obtained with experimentation using 30 dimensions, with this size of dimensions, the algorithm has a wider search space. In addition, the functions that are being optimized are complex, despite this situation, the method showed that it was approaching the optimum of each function, and the values obtained with both types of fuzzy logic were also very close.

Table 4.43 Dynamic Stochastic Fractal Search DSFS results with 30 dimensions

Dynamic Stochastic Fractal Search DFSF with 30 Dimensions					
Function	fi	Type 1 Fuzzy logic		Type 2 Fuzzy logic	
		Main	Std	Main	Std
f1	100	$3.49 \times 10^{+03}$	$2.76 \times 10^{+03}$	$3.24 \times 10^{+03}$	$2.74 \times 10^{+03}$
f2	200	$3.06 \times 10^{+16}$	$7.11 \times 10^{+16}$	$7.59 \times 10^{+17}$	$3.98 \times 10^{+18}$
f3	300	$8.40 \times 10^{+03}$	$3.95 \times 10^{+03}$	$8.79 \times 10^{+03}$	$4.60 \times 10^{+03}$
f4	400	4.87 $\times 10^{+02}$	$3.56 \times 10^{+01}$	4.80 $\times 10^{+02}$	$3.39 \times 10^{+01}$
f5	500	6.11 $\times 10^{+02}$	$2.27 \times 10^{+01}$	6.00 $\times 10^{+02}$	$2.02 \times 10^{+01}$
f6	600	6.00 $\times 10^{+02}$	1.36×10^{-02}	6.00 $\times 10^{+02}$	1.13×10^{-02}
f7	700	8.53 $\times 10^{+02}$	$1.68 \times 10^{+01}$	8.60 $\times 10^{+02}$	$1.70 \times 10^{+01}$
f8	800	9.05 $\times 10^{+02}$	$2.12 \times 10^{+01}$	9.06 $\times 10^{+02}$	$2.27 \times 10^{+01}$

f9	900	9.01 $\times 10^{+02}$	6.89×10^{-01}	9.04 $\times 10^{+02}$	$1.44 \times 10^{+01}$
f10	1000	$6.21 \times 10^{+03}$	6.89×10^{-01}	$6.05 \times 10^{+03}$	$5.03 \times 10^{+02}$
f11	1100	1.19 $\times 10^{+03}$	$2.31 \times 10^{+01}$	1.19 $\times 10^{+03}$	$2.90 \times 10^{+01}$
f12	1200	$1.56 \times 10^{+05}$	$1.03 \times 10^{+05}$	$1.81 \times 10^{+05}$	$1.48 \times 10^{+05}$
f13	1300	3.56 $\times 10^{+03}$	$9.09 \times 10^{+02}$	4.01 $\times 10^{+03}$	$1.00 \times 10^{+03}$
f14	1400	1.50 $\times 10^{+03}$	$1.03 \times 10^{+01}$	1.50 $\times 10^{+03}$	$1.06 \times 10^{+01}$
f15	1500	1.70 $\times 10^{+03}$	$4.43 \times 10^{+01}$	1.70 $\times 10^{+03}$	$3.95 \times 10^{+01}$
f16	1600	2.45 $\times 10^{+03}$	$2.66 \times 10^{+02}$	2.47 $\times 10^{+03}$	$2.14 \times 10^{+02}$
f17	1700	1.85 $\times 10^{+03}$	$6.65 \times 10^{+01}$	1.86 $\times 10^{+03}$	$8.06 \times 10^{+01}$
f18	1800	2.87 $\times 10^{+03}$	$6.86 \times 10^{+02}$	2.69 $\times 10^{+03}$	$3.69 \times 10^{+02}$
f19	1900	1.99 $\times 10^{+03}$	$1.88 \times 10^{+01}$	1.99 $\times 10^{+03}$	$1.87 \times 10^{+01}$
f20	2000	2.39 $\times 10^{+03}$	$2.58 \times 10^{+01}$	2.22 $\times 10^{+03}$	$1.12 \times 10^{+02}$
f21	2100	2.40 $\times 10^{+03}$	$2.28 \times 10^{+01}$	2.39 $\times 10^{+03}$	$2.51 \times 10^{+01}$
f22	2200	2.30 $\times 10^{+03}$	1.50×10^{-02}	2.30 $\times 10^{+03}$	7.44×10^{-03}
f23	2300	2.74 $\times 10^{+03}$	$2.86 \times 10^{+01}$	2.73 $\times 10^{+03}$	$2.26 \times 10^{+01}$
f24	2400	2.91 $\times 10^{+03}$	$3.37 \times 10^{+01}$	2.90 $\times 10^{+03}$	$3.04 \times 10^{+01}$
f25	2500	2.89 $\times 10^{+03}$	$1.73 \times 10^{+00}$	2.89 $\times 10^{+03}$	9.76×10^{-01}
f26	2600	4.28 $\times 10^{+03}$	$5.50 \times 10^{+02}$	4.22 $\times 10^{+03}$	$6.59 \times 10^{+02}$
f27	2700	3.22 $\times 10^{+03}$	$8.08 \times 10^{+00}$	3.22 $\times 10^{+03}$	$7.37 \times 10^{+00}$
f28	2800	3.21 $\times 10^{+03}$	$1.25 \times 10^{+01}$	3.21 $\times 10^{+03}$	$1.27 \times 10^{+01}$
f29	2900	3.63 $\times 10^{+03}$	$1.06 \times 10^{+02}$	3.59 $\times 10^{+03}$	$1.13 \times 10^{+02}$
f30	3000	1.32 $\times 10^{+04}$	$3.29 \times 10^{+03}$	1.51 $\times 10^{+04}$	$5.55 \times 10^{+03}$

Table 4.62 Dynamic Stochastic Fractal Search DSFS results with 50 dimensions

Dynamic Stochastic Fractal Search DSFS with 50 Dimensions					
Function	f_i	Type 1 Fuzzy logic		Type 2 Fuzzy logic	
		Main	Std	Main	Std
f1	100	$9.00 \times 10^{+04}$	$4.30 \times 10^{+04}$	$8.42 \times 10^{+04}$	$6.69 \times 10^{+04}$
f2	200	$1.20 \times 10^{+40}$	$4.53 \times 10^{+40}$	$5.08 \times 10^{+39}$	$1.52 \times 10^{+40}$
f3	300	$6.01 \times 10^{+04}$	$9.72 \times 10^{+03}$	$6.28 \times 10^{+04}$	$1.39 \times 10^{+04}$
f4	400	5.72 $\times 10^{+02}$	$4.05 \times 10^{+01}$	5.69 $\times 10^{+02}$	$4.26 \times 10^{+01}$
f5	500	7.68 $\times 10^{+02}$	$4.35 \times 10^{+01}$	7.69 $\times 10^{+02}$	$4.59 \times 10^{+01}$
f6	600	6.01 $\times 10^{+02}$	1.35×10^{-01}	6.01 $\times 10^{+02}$	1.55×10^{-01}
f7	700	$1.05 \times 10^{+03}$	$3.18 \times 10^{+01}$	$1.06 \times 10^{+03}$	$2.61 \times 10^{+01}$
f8	800	$1.06 \times 10^{+03}$	$3.69 \times 10^{+01}$	$1.06 \times 10^{+03}$	$4.42 \times 10^{+01}$
f9	900	$1.13 \times 10^{+03}$	$1.49 \times 10^{+02}$	$1.14 \times 10^{+03}$	$1.22 \times 10^{+02}$
f10	1000	$1.13 \times 10^{+04}$	$4.84 \times 10^{+02}$	$1.12 \times 10^{+04}$	$6.53 \times 10^{+02}$
f11	1100	1.36 $\times 10^{+03}$	$3.00 \times 10^{+01}$	1.36 $\times 10^{+03}$	$4.30 \times 10^{+01}$
f12	1200	$3.54 \times 10^{+06}$	$1.42 \times 10^{+06}$	$3.47 \times 10^{+06}$	$1.96 \times 10^{+06}$
f13	1300	$1.48 \times 10^{+04}$	$1.17 \times 10^{+04}$	$1.67 \times 10^{+04}$	$1.36 \times 10^{+04}$
f14	1400	1.50 $\times 10^{+03}$	$1.03 \times 10^{+01}$	1.80 $\times 10^{+03}$	$8.36 \times 10^{+01}$

f15	1500	4.14 $\times 10^{+03}$	$1.54 \times 10^{+03}$	3.75 $\times 10^{+03}$	$8.43 \times 10^{+02}$
f16	1600	3.52 $\times 10^{+03}$	$3.87 \times 10^{+02}$	3.58 $\times 10^{+03}$	$4.80 \times 10^{+02}$
f17	1700	3.05 $\times 10^{+03}$	$2.37 \times 10^{+02}$	3.04 $\times 10^{+03}$	$2.56 \times 10^{+02}$
f18	1800	$5.42 \times 10^{+04}$	$3.34 \times 10^{+04}$	$5.39 \times 10^{+04}$	$3.20 \times 10^{+04}$
f19	1900	7.10 $\times 10^{+03}$	$4.05 \times 10^{+03}$	7.73 $\times 10^{+03}$	$5.45 \times 10^{+03}$
f20	2000	3.10 $\times 10^{+03}$	$2.16 \times 10^{+02}$	3.14 $\times 10^{+03}$	$2.36 \times 10^{+02}$
f21	2100	2.55 $\times 10^{+03}$	$4.55 \times 10^{+01}$	2.55 $\times 10^{+03}$	$4.99 \times 10^{+01}$
f22	2200	$1.10 \times 10^{+04}$	$4.12 \times 10^{+03}$	$1.23 \times 10^{+04}$	$3.40 \times 10^{+03}$
f23	2300	3.00 $\times 10^{+03}$	$4.43 \times 10^{+01}$	2.98 $\times 10^{+03}$	$4.91 \times 10^{+01}$
f24	2400	3.14 $\times 10^{+03}$	$5.98 \times 10^{+01}$	3.14 $\times 10^{+03}$	$5.85 \times 10^{+01}$
f25	2500	3.07 $\times 10^{+03}$	2.52×10^{-01}	3.08 $\times 10^{+03}$	$2.22 \times 10^{+01}$
f26	2600	6.07 $\times 10^{+03}$	$5.92 \times 10^{+02}$	6.16 $\times 10^{+03}$	$4.77 \times 10^{+02}$
f27	2700	3.41 $\times 10^{+03}$	$4.40 \times 10^{+01}$	3.40 $\times 10^{+03}$	$4.09 \times 10^{+01}$
f28	2800	3.36 $\times 10^{+03}$	$3.36 \times 10^{+01}$	3.35 $\times 10^{+03}$	$3.44 \times 10^{+01}$
f29	2900	4.15 $\times 10^{+03}$	$2.71 \times 10^{+02}$	4.17 $\times 10^{+03}$	$2.55 \times 10^{+03}$
f30	3000	$3.25 \times 10^{+06}$	$6.84 \times 10^{+05}$	$3.24 \times 10^{+06}$	$8.20 \times 10^{+05}$

Table 4.44 Dynamic Stochastic Fractal Search DSFS results with 100 dimensions

Dynamic Stochastic Fractal Search DSFS with 100 Dimensions					
Function	fi	Type-1 Fuzzy logic		Type-2 Fuzzy logic	
		Main	Std	Main	Std
f1	100	$1.15 \times 10^{+08}$	$4.36 \times 10^{+07}$	$1.08 \times 10^{+08}$	$3.31 \times 10^{+07}$
f2	200	$1.34 \times 10^{+108}$	$8.85 \times 10^{+108}$	$1.14 \times 10^{+111}$	$5.81 \times 10^{+111}$
f3	300	$2.53 \times 10^{+05}$	$2.78 \times 10^{+04}$	$2.56 \times 10^{+05}$	$2.78 \times 10^{+04}$
f4	400	9.23 $\times 10^{+02}$	$4.74 \times 10^{+01}$	9.37 $\times 10^{+02}$	$6.10 \times 10^{+01}$
f5	500	$1.29 \times 10^{+03}$	$9.15 \times 10^{+01}$	$1.29 \times 10^{+03}$	$7.11 \times 10^{+01}$
f6	600	6.07 $\times 10^{+02}$	$1.33 \times 10^{+00}$	6.07 $\times 10^{+02}$	$1.02 \times 10^{+00}$
f7	700	$1.72 \times 10^{+03}$	$5.61 \times 10^{+01}$	$1.73 \times 10^{+03}$	$5.30 \times 10^{+01}$
f8	800	$1.58 \times 10^{+03}$	$7.70 \times 10^{+01}$	$1.59 \times 10^{+03}$	$8.69 \times 10^{+01}$
f9	900	$1.19 \times 10^{+04}$	$3.07 \times 10^{+03}$	$1.28 \times 10^{+04}$	$4.21 \times 10^{+03}$
f10	1000	$2.71 \times 10^{+04}$	$1.04 \times 10^{+03}$	$2.72 \times 10^{+04}$	$9.91 \times 10^{+02}$
f11	1100	$1.62 \times 10^{+04}$	$3.86 \times 10^{+03}$	$1.63 \times 10^{+04}$	$4.35 \times 10^{+03}$
f12	1200	$6.66 \times 10^{+07}$	$2.09 \times 10^{+07}$	$7.05 \times 10^{+07}$	$1.74 \times 10^{+07}$
f13	1300	5.66 $\times 10^{+03}$	$1.80 \times 10^{+03}$	6.16 $\times 10^{+03}$	$2.89 \times 10^{+03}$
f14	1400	$3.36 \times 10^{+05}$	$2.50 \times 10^{+05}$	$2.36 \times 10^{+05}$	$1.44 \times 10^{+05}$
f15	1500	4.46 $\times 10^{+03}$	$4.66 \times 10^{+03}$	4.39 $\times 10^{+03}$	$3.13 \times 10^{+03}$
f16	1600	7.97 $\times 10^{+03}$	$8.58 \times 10^{+02}$	7.70 $\times 10^{+03}$	$7.93 \times 10^{+02}$
f17	1700	6.04 $\times 10^{+03}$	$3.39 \times 10^{+02}$	5.90 $\times 10^{+03}$	$5.76 \times 10^{+02}$
f18	1800	$5.97 \times 10^{+05}$	$3.86 \times 10^{+05}$	$6.45 \times 10^{+05}$	$3.99 \times 10^{+05}$

f19	1900	$3.60 \times 10^{+03}$	$1.65 \times 10^{+03}$	$3.49 \times 10^{+03}$	$1.56 \times 10^{+03}$
f20	2000	$2.18 \times 10^{+03}$	$8.28 \times 10^{+01}$	$6.27 \times 10^{+03}$	$4.30 \times 10^{+02}$
f21	2100	$3.11 \times 10^{+03}$	$6.75 \times 10^{+01}$	$3.11 \times 10^{+03}$	$6.06 \times 10^{+01}$
f22	2200	$2.96 \times 10^{+04}$	$8.67 \times 10^{+02}$	$2.97 \times 10^{+04}$	$8.23 \times 10^{+02}$
f23	2300	$3.59 \times 10^{+03}$	$5.6 \times 10^{+01}$	$3.57 \times 10^{+03}$	$7.87 \times 10^{+01}$
f24	2400	$4.09 \times 10^{+03}$	$9.76 \times 10^{+01}$	$4.08 \times 10^{+03}$	$1.19 \times 10^{+02}$
f25	2500	$3.63 \times 10^{+03}$	$6.17 \times 10^{+01}$	$3.63 \times 10^{+03}$	$5.46 \times 10^{+01}$
f26	2600	$1.41 \times 10^{+04}$	$1.04 \times 10^{+03}$	$1.42 \times 10^{+04}$	$8.28 \times 10^{+02}$
f27	2700	$3.72 \times 10^{+03}$	$7.06 \times 10^{+01}$	$3.73 \times 10^{+03}$	$5.89 \times 10^{+01}$
f28	2800	$3.96 \times 10^{+03}$	$1.11 \times 10^{+02}$	$3.96 \times 10^{+03}$	$1.35 \times 10^{+02}$
f29	2900	$7.96 \times 10^{+03}$	$5.34 \times 10^{+02}$	$7.96 \times 10^{+03}$	$5.11 \times 10^{+02}$
f30	3000	$3.04 \times 10^{+05}$	$1.27 \times 10^{+05}$	$3.18 \times 10^{+05}$	$1.68 \times 10^{+05}$

Multimodal functions are more difficult to optimize than unimodal ones due to the complexity that they represent, because the algorithms must escape or avoid local optima and arrive to the global optimal solution. In this study, not only are unimodal and multimodal functions being optimized, but also hybrid and complex functions are being optimized using different values of dimensions, as can be seen in Tables 4.62 and 4.63 with 50 and 100 dimensions respectively. In addition, the values obtained with the variants using Type-1 and Type-2 fuzzy systems for adaptation of parameters, show that the method had some degree of difficulty in reaching the global optimum, even so, there provide good approximation values showing that the improved method is efficient in optimization tasks.

In the literature, we can find the hybrid optimization algorithm of fireflies and swarm of particles to solve costly computational problems [57] in the optimization of the functions of CEC'2017, its performance was compared with the combination of the hybrid algorithm of firefly and optimization of swarm of particles (HFPSO). The experimentation was carried out with the following specifications: 20 independent runs, for each case the maximum number of evaluations of 500 for 10 dimensions (10D) and 1500 for 30-dimensional problems (30D) was used.

Table 4.45 HFPSO vs DSFS results with 10 dimensions

Function	HFPSO [57]			DSFS		DSFS	
	fi	Mean	Std	Type 1 fuzzy logic		Type 2 fuzzy logic	
				Mean	Std	Mean	Std
f1	100	$9.81 \times 10^{+08}$	$1.01 \times 10^{+02}$	$1.01 \times 10^{+02}$	4.87×10^{-01}	$1.01 \times 10^{+02}$	7.85×10^{-01}
f2	200	$4.91 \times 10^{+08}$	$2.00 \times 10^{+02}$	$2.00 \times 10^{+02}$	$0.00 \times 10^{+00}$	$2.00 \times 10^{+02}$	$0.00 \times 10^{+00}$

f3	300	5.96x10 ⁺⁰³	3.00x10 ⁺⁰²	3.00x10⁺⁰²	3.73x10 ⁺⁰⁶	3.00x10⁺⁰²	5.04x10 ⁻⁰⁶
f4	400	4.55x10 ⁺⁰¹	4.01x10 ⁺⁰²	4.01x10⁺⁰²	7.81x10 ⁺⁰¹	4.00x10⁺⁰²	6.19x10 ⁺⁰¹
f5	500	1.84x10 ⁺⁰¹	5.06x10 ⁺⁰²	5.06x10⁺⁰²	2.04x10 ⁺⁰⁰	5.07x10⁺⁰²	2.64x10 ⁺⁰⁰
f6	600	1.35x10 ⁺⁰¹	6.00x10 ⁺⁰²	6.00x10⁺⁰²	7.21x10 ⁻⁰⁸	6.00x10⁺⁰²	6.24x10 ⁻⁰⁸
f7	700	1.73x10 ⁺⁰¹	7.19x10 ⁺⁰²	7.19x10⁺⁰²	3.45x10 ⁺⁰⁰	7.20x10⁺⁰²	3.32x10 ⁺⁰⁰
f8	800	1.44x10 ⁺⁰¹	8.07x10 ⁺⁰²	8.07x10⁺⁰²	3.04x10 ⁺⁰⁰	8.07x10⁺⁰²	2.44x10 ⁺⁰⁰
f9	900	3.07x10 ⁺⁰²	9.00x10 ⁺⁰²	9.00x10⁺⁰²	0.00x10 ⁺⁰⁰	9.00x10⁺⁰²	0.00x10 ⁺⁰⁰
f10	100	3.79x10 ⁺⁰²	1.40x10 ⁺⁰³	1.40x10⁺⁰³	1.70x10 ⁺⁰²	1.34x10⁺⁰³	1.31x10 ⁺⁰²
f11	110	5.24x10 ⁺⁰¹	1.10x10 ⁺⁰³	1.10x10⁺⁰³	9.31x10 ⁻⁰¹	1.10x10⁺⁰³	1.00x10 ⁺⁰⁰
f12	120	4.13x10 ⁺⁰⁶	1.50x10 ⁺⁰³	1.50x10⁺⁰³	1.04x10 ⁺⁰²	1.52x10⁺⁰³	1.02x10 ⁺⁰²
f13	130	7.68x10 ⁺⁰³	1.31x10 ⁺⁰³	1.31x10⁺⁰³	4.03x10 ⁺⁰⁰	1.31x10⁺⁰³	4.14x10 ⁺⁰⁰
f14	140	4.18x10 ⁺⁰³	1.40x10 ⁺⁰³	1.40x10⁺⁰³	1.69x10 ⁺⁰⁰	1.40x10⁺⁰³	2.01x10 ⁺⁰⁰
f15	150	2.37x10 ⁺⁰⁴	1.50x10 ⁺⁰³	1.50E+03	7.24x10 ⁺⁰¹	1.50x10⁺⁰³	8.88x10 ⁺⁰¹
f16	160	1.59x10 ⁺⁰²	1.60x10 ⁺⁰³	1.60x10⁺⁰³	2.79x10 ⁺⁰¹	1.60x10⁺⁰³	3.64x10 ⁺⁰¹
f17	170	8.40x10 ⁺⁰¹	1.70x10 ⁺⁰³	1.70x10⁺⁰³	2.68x10 ⁺⁰⁰	1.71x10⁺⁰³	4.41x10 ⁺⁰⁰
f18	180	1.79x10 ⁺⁰⁴	1.81x10 ⁺⁰³	1.81x10⁺⁰³	2.56x10 ⁺⁰⁰	1.81x10⁺⁰³	2.57x10 ⁺⁰⁰
f19	190	3.83x10 ⁺⁰⁴	1.90x10 ⁺⁰³	1.90x10⁺⁰³	4.15x10 ⁺⁰¹	1.9x10⁺⁰³	4.45x10 ⁺⁰¹
f20	200	1.08x10 ⁺⁰²	2.00x10 ⁺⁰³	2.00x10⁺⁰³	1.39x10 ⁺⁰¹	2.00x10⁺⁰³	4.73x10 ⁻⁰³
f21	210	4.78x10 ⁺⁰¹	2.23x10 ⁺⁰³	2.23x10⁺⁰³	5.08x10 ⁺⁰¹	2.24x10⁺⁰³	5.40x10 ⁺⁰¹
f22	220	5.88x10 ⁺⁰²	2.29x10 ⁺⁰³	2.29x10⁺⁰³	2.17x10 ⁺⁰¹	2.28x10⁺⁰³	3.93x10 ⁺⁰¹
f23	230	2.87x10 ⁺⁰¹	2.61x10 ⁺⁰³	2.61x10⁺⁰³	2.80x10 ⁺⁰⁰	2.60x10⁺⁰³	5.61x10 ⁺⁰¹
f24	240	1.47x10 ⁺⁰²	2.61x10 ⁺⁰³	2.61x10⁺⁰³	1.22x10 ⁺⁰²	2.63x10⁺⁰³	1.14x10 ⁺⁰²
f25	250	5.02x10 ⁺⁰¹	2.90x10 ⁺⁰³	2.90x10⁺⁰³	1.08x10 ⁺⁰¹	2.90x10⁺⁰³	1.15x10 ⁺⁰¹
f26	260	3.42x10 ⁺⁰²	2.90x10 ⁺⁰³	2.90x10⁺⁰³	2.15x10 ⁺⁰¹	2.90x10⁺⁰³	3.02x10 ⁺⁰¹
f27	270	3.94x10 ⁺⁰²	3.09x10 ⁺⁰³	3.09x10⁺⁰³	1.97x10 ⁺⁰⁰	3.09x10⁺⁰³	2.14x10 ⁺⁰⁰
f28	280	1.08x10 ⁺⁰²	3.09x10 ⁺⁰³	3.09x10⁺⁰³	4.20x10 ⁺⁰¹	3.11x10⁺⁰³	5.69x10 ⁺⁰¹
f29	290	9.40x10 ⁺⁰¹	3.16x10 ⁺⁰³	3.16x10⁺⁰³	1.01x10 ⁺⁰¹	3.16x10⁺⁰³	9.65x10 ⁺⁰⁰
f30	300	3.75x10 ⁺⁰⁶	3.56x10 ⁺⁰³	3.56x10⁺⁰³	2.30E+02	3.55x10⁺⁰³	1.19x10 ⁺⁰²

Funcio n	fi	HFPSO [57]		DSFS Type 1 fuzzy logic		DSFS Type 2 fuzzy logic	
		Mean	Std	Mean	Std	Mean	Std
f1	100	9.81E+08	1.01E+02	3.49x10 ⁺⁰³	2.76x10 ⁺⁰³	3.24x10 ⁺⁰³	2.74x10 ⁺⁰³
f2	200	4.91E+08	2.00E+02	3.06E+16	7.11E+16	7.59E+17	3.98E+18
f3	300	5.96E+03	3.00E+02	8.40x10 ⁺⁰³	3.95x10 ⁺⁰³	8.79x10 ⁺⁰³	4.60x10 ⁺⁰³
f4	400	4.55E+01	4.01E+02	4.87E+02	3.56E+01	4.80E+02	3.39E+01
f5	500	1.84E+01	5.06E+02	6.11E+02	2.27E+01	6.00E+02	2.02E+01
f6	600	1.35E+01	6.00E+02	6.00E+02	1.36E-02	6.00E+02	1.13E-02
f7	700	1.73E+01	7.19E+02	8.53E+02	1.68E+01	8.60E+02	1.70E+01
f8	800	1.44E+01	8.07E+02	9.05E+02	2.12E+01	9.06E+02	2.27E+01
f9	900	3.07E+02	9.00E+02	9.01E+02	6.89E-01	9.04E+02	1.44E+01
f10	1000	3.79E+02	1.40E+03	6.21E+03	6.89E-01	6.05E+03	5.03E+02
f11	1100	5.24E+01	1.10E+03	1.19E+03	2.31E+01	1.19E+03	2.90E+01
f12	1200	4.13E+06	1.50E+03	1.56E+05	1.03E+05	1.81E+05	1.48E+05
f13	1300	7.68E+03	1.31E+03	3.56E+03	9.09E+02	4.01E+03	1.00E+03
f14	1400	4.18E+03	1.40E+03	1.50E+03	1.03E+01	1.50E+03	1.06E+01
f15	1500	2.37E+04	1.50E+03	1.70E+03	4.43E+01	1.70E+03	3.95E+01

f16	1600	1.59E+02	1.60E+03	2.45E+03	2.66E+02	2.47E+03	2.14E+02
f17	1700	8.40E+01	1.70E+03	1.85E+03	6.65E+01	1.86E+03	8.06E+01
f18	1800	1.79E+04	1.81E+03	2.87E+03	6.86E+02	2.69E+03	3.69E+02
f19	1900	3.83E+04	1.90E+03	1.99E+03	1.88E+01	1.99E+03	1.87E+01
f20	2000	1.08E+02	2.00E+03	2.39E+03	2.58E+01	2.22E+03	1.12E+02
f21	2100	4.78E+01	2.23E+03	2.40E+03	2.28E+01	2.39E+03	2.51E+01
f22	2200	5.88E+02	2.29E+03	2.30E+03	1.50E-02	2.30E+03	7.44E-03
f23	2300	2.87E+01	2.61E+03	2.74E+03	2.86E+01	2.73E+03	2.26E+01
f24	2400	1.47E+02	2.61E+03	2.91E+03	3.37E+01	2.90E+03	3.04E+01
f25	2500	5.02E+01	2.90E+03	2.89E+03	1.73E+00	2.89E+03	9.76E-01
f26	2600	3.42E+02	2.90E+03	4.28E+03	5.50E+02	4.22E+03	6.59E+02
f27	2700	3.94E+01	3.09E+03	3.22E+03	8.08E+00	3.22E+03	7.37E+00
f28	2800	1.08E+02	3.09E+03	3.21E+03	1.25E+01	3.21E+03	1.27E+01
f29	2900	9.40E+01	3.16E+03	3.63E+03	1.06E+02	3.59E+03	1.13E+02
f30	3000	3.75E+06	3.56E+03	1.32E+04	3.29E+03	1.51E+04	5.55E+03

Table 4.46 HFPSO vs DSFS results with 30 dimensions

Tables 4.64 and 4.65 compare the results of the HFPSO with the DSFS proposed with both types of fuzzy logic. The experimentation of the CEC'2017 functions with 10D and 30 D respectively, show us that the dynamic stochastic fractal search (DSFS) method obtained on average better results in finding the global solutions of the functions. As previously mentioned, the results of dynamic stochastic fractal search (DSFS) with type-1 and type-2 fuzzy systems were very close, therefore, in comparison with HFPSO, better overall results were also obtained for each function.

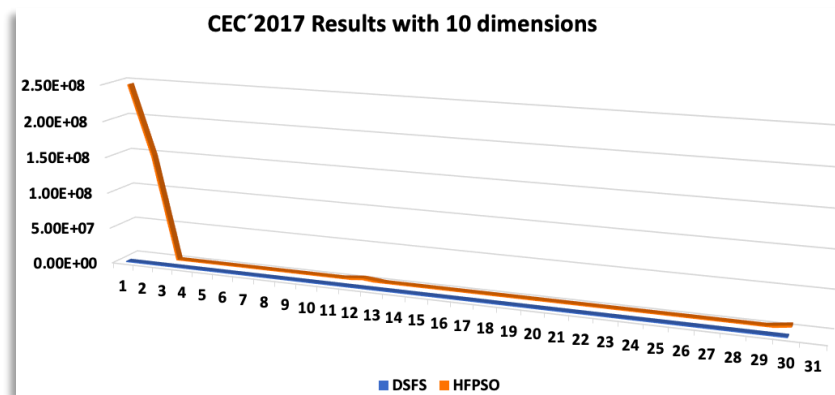


Fig. 4.44 DSFS vs HFPSO Results with 10 dimensions

On the x-axis we can find the values obtained with the DSFS and HFPSO methods, and the horizontal axis shows the average results of the evaluated functions. The DSFS approach, indicated by the blue sample line, has a better efficiency than the HFPSO orange line, because the values are closer to the global optima, as seen in Figure 4.56.

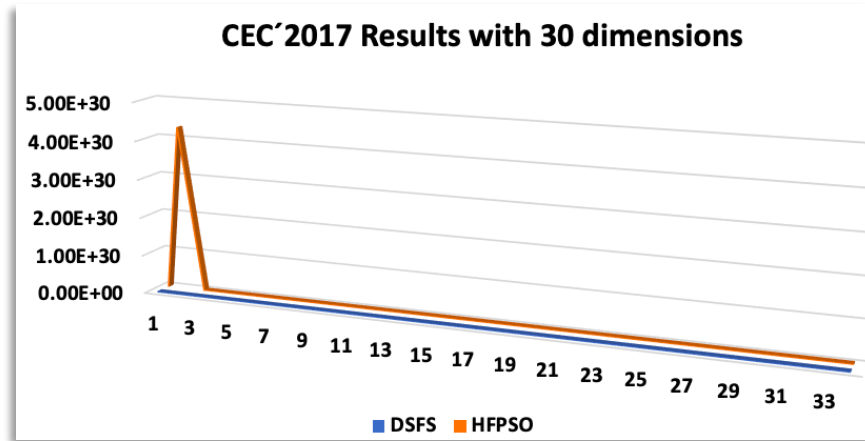


Fig. 4.45 Results of CEC'2017 with 30 dimensions

Fig. 4.57 illustrates the results obtained for the DSFS and HFPSO, using 30 dimensions, the vertical axis contains the values reached by the methods when evaluating the CEC'2017 functions and the horizontal axis shows the corresponding function. The orange line is the representative of the HFPSO algorithm where it is observed that in the first functions the values were very separate from the optimal ones, then for the f1, f2, f3, f4 functions the achieved performance can be viewed as unsatisfactory, with results up to 10^{+30} . On the other hand, the DFSF method obtained results closer to the global optimal being represented by the blue line, we will have to remember that the dimensions are high and therefore, the performance of the methods is not the same, to use lower dimensions where the efficiency of the algorithms is much better for each of the functions.

4.5.2 Z-test of HFPSO and DSFS Methods

To determine which of HFPSO or DSFS provided the closest to optimal result for each function, a statistical comparison was performed using the parametric z test. The formula for the Z test is expressed mathematically in the following fashion:

$$Z = \frac{(\bar{x}_1 - \bar{x}_2) - (\mu_1 - \mu_2)}{\sigma_{\bar{x}_1 - \bar{x}_2}} \quad (4.13)$$

where:

$\bar{x}_1 - \bar{x}_2$: It is the observed difference in the mean values of the methods.

$\mu_1 - \mu_2$: It is the expected difference in the mean values of the methods.

$\sigma_{\bar{x}_1 - \bar{x}_2}$: Standard error of the differences.

As can be seen in Table 4.66, column 7, the values obtained by the z-test, provide statistical evidence that the DSFS method using type-1 fuzzy logic is significantly better than HFPSO, in functions with 10 dimensions (Bold indicates best values).

Table 4.47 Z-test for 10 dimensions

Function	fi	HFPSO [58]		DSFS Type 1 fuzzy logic		z
		Mean	Std	Mean	Std	
f1	100	9.81E+08	1.01E+02	1.01E+02	4.87E-01	3.76E+07
f2	200	4.91E+08	2.00E+02	2.00E+02	0.00E+00	9.51E+06
f3	300	5.96E+03	3.00E+02	3.00E+02	3.73E-06	7.31E+01
f4	400	4.55E+01	4.01E+02	4.01E+02	7.81E-01	-3.43x10 ⁺⁰⁰
f5	500	1.84E+01	5.06E+02	5.06E+02	2.04E+00	-3.73x10 ⁺⁰⁰
f6	600	1.35E+01	6.00E+02	6.00E+02	7.21E-08	-3.79x10 ⁺⁰⁰
f7	700	1.73x10 ⁺⁰¹	7.19x10 ⁺⁰²	7.19x10⁺⁰²	3.45x10 ⁺⁰⁰	-3.78x10 ⁺⁰⁰
f8	800	1.44x10 ⁺⁰¹	8.07x10 ⁺⁰²	8.07x10⁺⁰²	3.04x10 ⁺⁰⁰	-3.80x10 ⁺⁰⁰
f9	900	3.07x10⁺⁰²	9.00x10 ⁺⁰²	9.00x10⁺⁰²	0.00x10 ⁺⁰⁰	-2.55x10 ⁺⁰⁰
f10	1000	3.79Ex10 ⁺⁰²	1.40x10 ⁺⁰³	1.40x10⁺⁰³	1.70x10 ⁺⁰²	-2.82x10 ⁺⁰⁰
f11	1100	5.24x10 ⁺⁰¹	1.10x10 ⁺⁰³	1.10x10⁺⁰³	9.31x10 ⁺⁰¹	-3.69x10 ⁺⁰⁰
f12	1200	4.1x10 ⁺⁰⁶	1.50x10 ⁺⁰³	1.50x10⁺⁰³	1.04x10 ⁺⁰²	1.07x10 ⁺⁰⁴
f13	1300	7.68x10⁺⁰³	1.31x10 ⁺⁰³	1.31x10⁺⁰³	4.03x10 ⁺⁰⁰	1.88x10 ⁺⁰¹
f14	1400	4.18x10⁺⁰³	1.40x10 ⁺⁰³	1.40x10⁺⁰³	1.69x10 ⁺⁰⁰	7.69x10 ⁺⁰⁰
f15	1500	2.37x10 ⁺⁰⁴	1.50x10 ⁺⁰³	1.50x10⁺⁰³	7.24x10 ⁺⁰¹	5.73E+01
f16	1600	1.59x10 ⁺⁰²	1.60x10 ⁺⁰³	1.60x10⁺⁰³	2.79x10 ⁺⁰¹	-3.49x10 ⁺⁰⁰
f17	1700	8.40x10 ⁺⁰¹	1.70x10 ⁺⁰³	1.70x10⁺⁰³	2.68x10 ⁺⁰⁰	-3.68x10 ⁺⁰⁰
f18	1800	1.79x10 ⁺⁰⁴	1.81x10 ⁺⁰³	1.81x10⁺⁰³	2.56x10 ⁺⁰⁰	3.44x10 ⁺⁰¹
f19	1900	3.83x10 ⁺⁰⁴	1.90x10 ⁺⁰³	1.90x10⁺⁰³	4.15x10 ⁺⁰¹	7.42x10 ⁺⁰¹
f20	2000	1.08x10 ⁺⁰²	2.00x10 ⁺⁰³	2.00x10⁺⁰³	1.39x10 ⁺⁰³	-3.66x10 ⁺⁰⁰
f21	2100	4.78x10 ⁺⁰¹	2.23x10 ⁺⁰³	2.23x10⁺⁰³	5.08x10 ⁺⁰¹	-3.79x10 ⁺⁰⁰

f22	2200	$5.88x10^{+02}$	$2.29x10^{+03}$	$2.29x10^{+03}$	$2.17x10^{+01}$	$-2.88x10^{+00}$
f23	2300	$2.87x10^{+01}$	$2.61x10^{+03}$	$2.61x10^{+03}$	$2.80x10^{+00}$	$-3.83x10^{+00}$
f24	2400	$1.47x10^{+02}$	$2.61x10^{+03}$	$2.61x10^{+03}$	$1.22x10^{+02}$	$-3.65x10^{+00}$
f25	2500	$5.02x10^{+01}$	$2.90x10^{+03}$	$2.90x10^{+03}$	$1.08x10^{+01}$	$-3.81x10^{+00}$
f26	2600	$3.42x10^{+02}$	$2.90x10^{+03}$	$2.90x10^{+03}$	$2.15x10^{+01}$	$-3.42x10^{+00}$
f27	2700	$3.94x10^{+01}$	$3.09x10^{+03}$	$3.09x10^{+03}$	$1.97x10^{+00}$	$-3.82x10^{+00}$
f28	2800	$1.08x10^{+02}$	$3.09x10^{+03}$	$3.09x10^{+03}$	$4.20x10^{+01}$	$-3.74x10^{+00}$
f29	2900	$9.40x10^{+01}$	$3.16x10^{+03}$	$3.16x10^{+03}$	$1.01x10^{+01}$	$-3.76x10^{+00}$
f30	3000	$3.75x10^{+06}$	$3.56x10^{+03}$	$3.56x10^{+03}$	$2.30x10^{+02}$	$4.08x10^{+03}$

Table 4.67 describes in row 2, column 7 the results when applying the z test to the methods compared to DSFS with fuzzy logic type 1 vs HFPSO using 30 dimensions. Again, the proposed method is significantly better.

Table 4.48 Results z-test for 30 dimensions

Function	fi	HFPSO [58]		DSFS		z
		Mean	Std	Mean	Std	
f1	100	$9.81x10^{+08}$	$1.01x10^{+02}$	$3.49x10^{+03}$	$2.76x10^{+03}$	$1.95x10^{+06}$
f2	200	$4.91x10^{+08}$	$2.00x10^{+02}$	$3.06x10^{+16}$	$7.11x10^{+16}$	$-2.36x10^{+00}$
f3	300	$5.96x10^{+03}$	$3.00x10^{+02}$	$8.40x10^{+03}$	$3.95x10^{+03}$	$-3.37x10^{+00}$
f4	400	$4.55x10^{+01}$	$4.01x10^{+02}$	$4.87x10^{+02}$	$3.56x10^{+01}$	$-6.01x10^{+00}$
f5	500	$1.84x10^{+01}$	$5.06x10^{+02}$	$6.11x10^{+02}$	$2.27x10^{+01}$	$-6.41x10^{+00}$
f6	600	$1.35x10^{+01}$	$6.00x10^{+02}$	$6.00x10^{+02}$	$1.36x10^{+02}$	$-5.35x10^{+00}$
f7	700	$1.73x10^{+01}$	$7.19x10^{+02}$	$8.53x10^{+02}$	$1.68x10^{+01}$	$-6.36x10^{+00}$
f8	800	$1.44x10^{+01}$	$8.07x10^{+02}$	$9.05x10^{+02}$	$2.12x10^{+01}$	$-6.04x10^{+00}$
f9	900	$3.07x10^{+02}$	$9.00x10^{+02}$	$9.01x10^{+02}$	$6.89x10^{+01}$	$-3.61x10^{+00}$
f10	1000	$3.79x10^{+02}$	$1.40x10^{+03}$	$6.21x10^{+03}$	$6.89x10^{+01}$	$-2.28x10^{+01}$
f11	1100	$5.24x10^{+01}$	$1.10E+03$	$1.19x10^{+03}$	$2.31x10^{+03}$	$-5.66x10^{+00}$
f12	1200	$4.13x10^{+06}$	$1.50E+03$	$1.56x10^{+05}$	$1.03x10^{+05}$	$2.11x10^{+02}$
f13	1300	$7.68x10^{+03}$	$1.31E+03$	$3.56x10^{+03}$	$9.09x10^{+02}$	$1.42x10^{+01}$
f14	1400	$4.18x10^{+03}$	$1.40E+03$	$1.50x10^{+03}$	$1.03x10^{+01}$	$1.05x10^{+01}$
f15	1500	$2.37x10^{+04}$	$1.50E+03$	$1.70x10^{+03}$	$4.43x10^{+01}$	$8.03x10^{+01}$
f16	1600	$1.59x10^{+02}$	$1.60E+03$	$2.45E+03$	$2.66x10^{+02}$	$-7.74x10^{+00}$
f17	1700	$8.40x10^{+01}$	$1.70x10^{+03}$	$1.85x10^{+03}$	$6.65x10^{+01}$	$-5.69x10^{+00}$
f18	1800	$1.79x10^{+04}$	$1.81x10^{+03}$	$2.87x10^{+03}$	$6.86x10^{+02}$	$4.25x10^{+01}$
f19	1900	$3.83x10^{+04}$	$1.90x10^{+03}$	$1.99x10^{+03}$	$1.88x10^{+01}$	$1.05x10^{+02}$
f20	2000	$1.08x10^{+02}$	$2.00x10^{+03}$	$2.39x10^{+03}$	$2.58x10^{+01}$	$-6.25x10^{+00}$

f21	2100	4.78E+01	2.23E+03	2.40x10⁺⁰³	2.28x10 ⁺⁰¹	-5.78x10 ⁺⁰⁰
f22	2200	5.88x10 ⁺⁰²	2.29x10 ⁺⁰³	2.30x10⁺⁰³	1.50x10 ⁺⁰²	-4.09x10 ⁺⁰⁰
f23	2300	2.87x10 ⁺⁰¹	2.61x10 ⁺⁰³	2.74x10⁺⁰³	2.86x10 ⁺⁰¹	-5.69x10 ⁺⁰⁰
f24	2400	1.47x10 ⁺⁰²	2.61x10 ⁺⁰³	2.91x10⁺⁰³	3.37x10 ⁺⁰¹	-5.80x10 ⁺⁰⁰
f25	2500	5.02x10 ⁺⁰¹	2.90x10 ⁺⁰³	2.89x10⁺⁰³	1.73x10 ⁺⁰⁰	-5.36x10 ⁺⁰⁰
f26	2600	3.42x10 ⁺⁰²	2.90x10 ⁺⁰³	4.28x10 ⁺⁰³	5.50x10 ⁺⁰²	-7.31x10 ⁺⁰⁰
f27	2700	3.94x10 ⁺⁰¹	3.09x10 ⁺⁰³	3.22x10 ⁺⁰³	8.08x10 ⁺⁰⁰	-5.64x10 ⁺⁰⁰
f28	2800	1.08x10 ⁺⁰²	3.09x10 ⁺⁰³	3.21x10 ⁺⁰³	1.25x10 ⁺⁰¹	-5.50x10 ⁺⁰⁰
f29	2900	9.40x10 ⁺⁰¹	3.16x10 ⁺⁰³	3.63x10 ⁺⁰³	1.06x10 ⁺⁰²	-6.13x10 ⁺⁰⁰
f30	3000	3.75x10 ⁺⁰⁶	3.56x10 ⁺⁰³	1.32x10 ⁺⁰⁴	3.29x10 ⁺⁰³	4.22x10 ⁺⁰³

4.6 Optimization a type 1 and type 2 fuzzy controller

In the fifth external case study, the optimization of a fuzzy controller with type 1 and type 2 fuzzy logic is performed, the methodology of which is described below [58]:

4.6.1 Proposed method for optimizing of a type 1 and type 2 fuzzy logic controller

The main contribution of this work is the proposed optimization methodology, as well as its application for finding the optimal parameters in the design of a type-2 fuzzy controller, which must control the behavior of an autonomous mobile robot. The fuzzy controller aims at following a given trajectory with a margin of error as minimum as possible, calculated by means of an established metric of control performance.

The most important difference between using the dynamic adjustment of parameters in metaheuristic algorithms with respect to the use of fixed parameters is that the parameters selected for dynamic adjustment are modified as the iterations proceed, which leads us to obtain better solutions.

The methodology consists on using a metaheuristic algorithm to perform the optimization of the membership functions parameters of the type-2 fuzzy robot controller for path following. In this case, we measure the type-2 fuzzy controller's performance and return a result according to the established performance metric and continue with the optimization until a stopping criterion or a previously established number of iterations are met. This methodology can be found in a summarized form in Fig. 4.58.

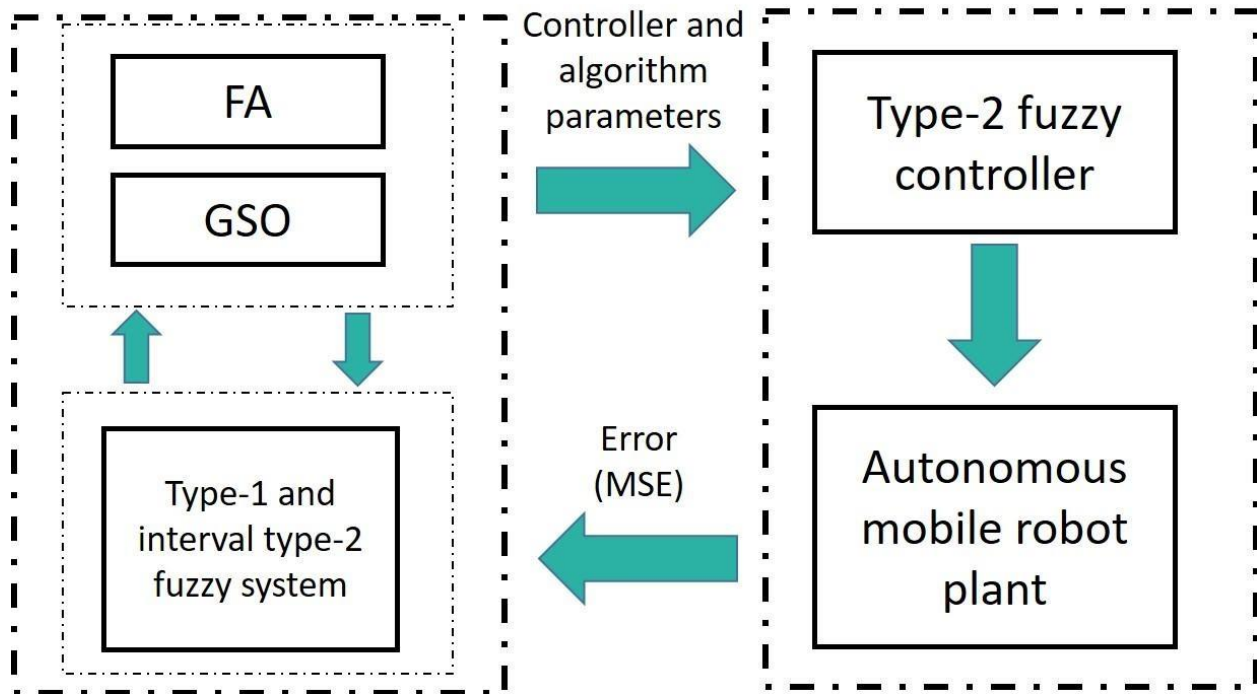


Fig. 4.46 Proposed optimization method

The optimization of the fuzzy controller is performed with the original algorithms and their variants using dynamic parameter adjustment in the GSO and FA algorithms. In the first case, for the optimization variants we use type-1 fuzzy systems. The fuzzy system used for parameter adjustment of the GSO algorithm uses "Iteration" as input variable and as output variables the c_3 and c_4 parameters. Each of the variables is composed of three triangular membership functions labelled as "low", "medium" and "high", as can be seen in Figure 4.59. The fuzzy system FA_T1 that performs parameter adjustment of the FA uses the "Iteration" variable as input variable and as an output variable the parameter, and these variables are composed of three triangular membership functions labelled as "low", "medium" and "high", as illustrated in Fig. 4.60.

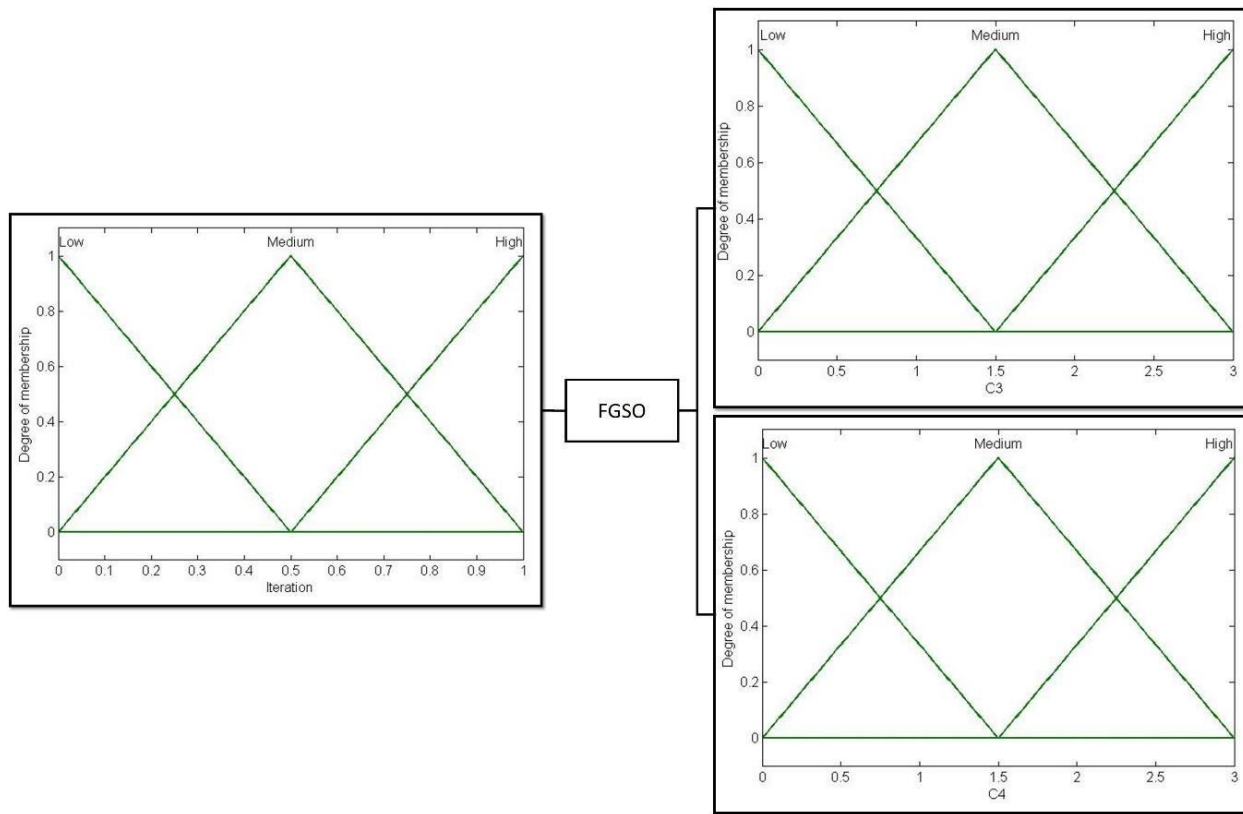


Fig. 4.47 Type-1 fuzzy system for the GSO algorithm

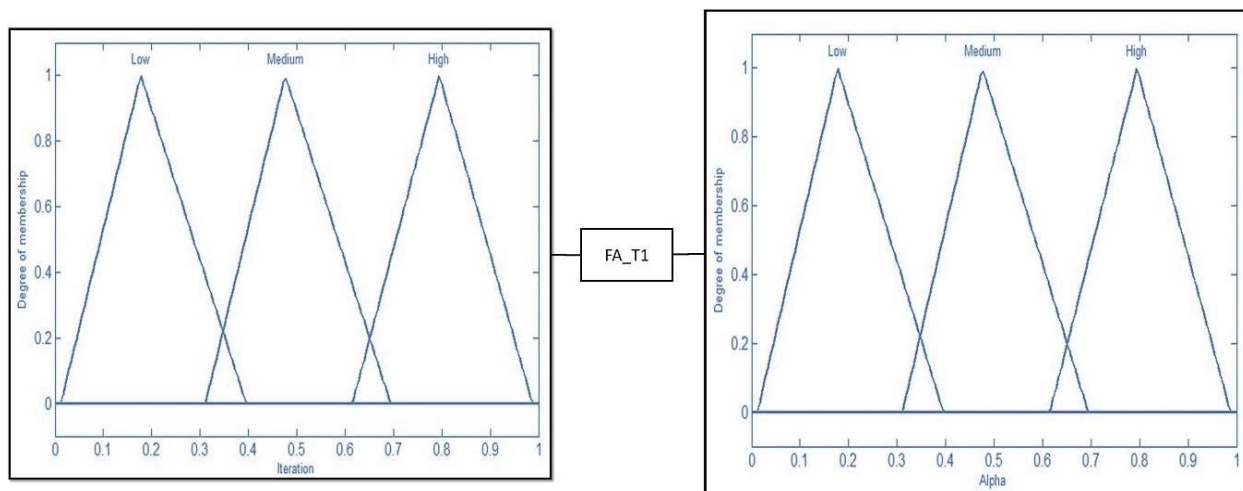


Fig. 4.48 Type-1 fuzzy system for the FA

In the second variant, interval type-2 fuzzy systems are used in FGSO IT2 and FA_T2, which are an extension of the previously used type-1 fuzzy systems. The type-2 fuzzy systems are formed with the same input and output variables, which are partitioned in the same way and sharing

the same form the set of fuzzy rules. This fuzzy system can be found in Fig. 4.61 and 4.62 as shown below, respectively:

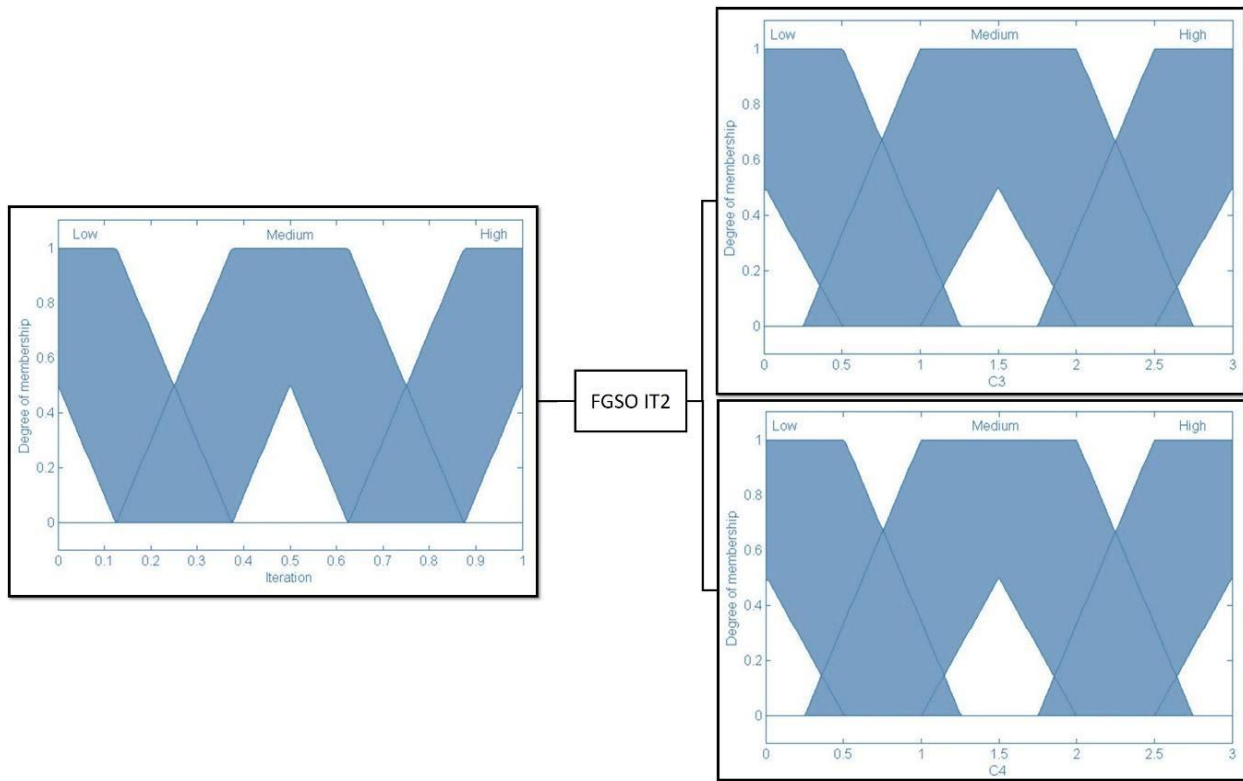


Fig. 4.49 Interval type-2 fuzzy system for the GSO algorithm

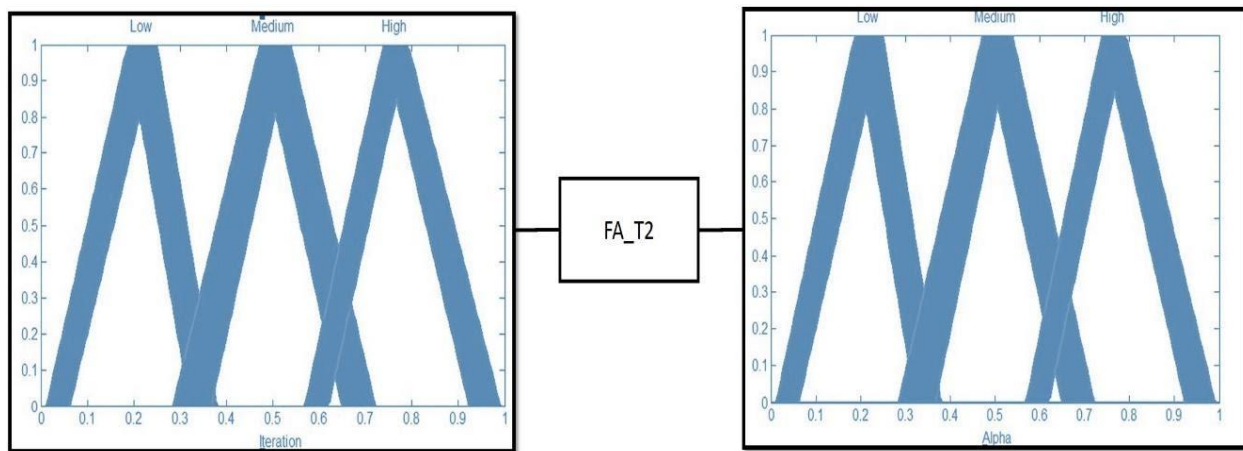


Fig. 4.50 Interval type-2 fuzzy system for the FA

Fuzzy system rules were designed with the idea that when the algorithms are in the initial iterations they can explore and when they are in the last iterations they can exploit.

Table 4.49 Fuzzy rules for FGSO

Fuzzy rules for FGSO
1. If (Iteration is Low) then (c3 is Low) and (c4 is High)
2. If (Iteration is Medium) then (c3 is Medium) and (c4 is Medium)
3. If (Iteration is High) then (c3 is High) and (c4 is Low)

Table 4.50 Fuzzy rules for FA

Fuzzy rules for FA
1.- if (Iteration is Low) then (Alpha is Low)
2.- if (Iteration is Medium) then (Alpha is Medium)
3.- if (Iteration is High) then (Alpha is High)

Each individual or possible solution that is used in our proposal is formed by the required parameters to automatically generate the type-2 fuzzy controller avoiding manually generating the fuzzy controller, in this way expecting to obtain better results than those obtained using the basis fuzzy controller. In Figure 4.63 we can find the way in which individuals are composed in the metaheuristic algorithms [59], [60].

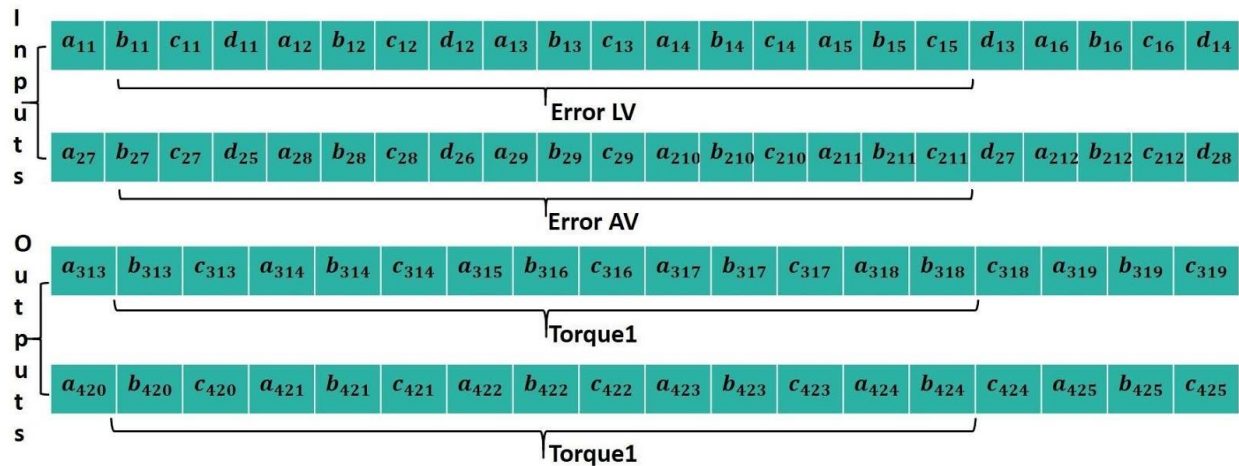


Fig. 4.51 Structure of individuals in the GSO algorithm and FA

Each individual contains the necessary parameters to form the fuzzy controller. In this case, the controller has two input variables, which are formed by two trapezoidal membership functions

and this correspond to 8 parameters to form each and a triangular with six parameters giving a total of 22 parameters per input variable. The output variables consist of three triangular membership functions which are formed with six parameters each, giving a total of 18 parameters per output variable. Once all the parameters are obtained to form the input and output variables, the fuzzy controller is built.

The structure of the individuals in the GSO and FA algorithms help us to form the interval type-2 fuzzy controllers, which are formed by triangular and trapezoidal membership functions, and their equations and graphical representation using type-2 fuzzy logic are presented below. The representation of the interval type-2 triangular membership functions (itritype2) of the fuzzy controller consists of 6 parameters a_{n1} , b_{n1} , c_{n1} , a_{n2} , b_{n2} and c_{n2} , where the parameter values satisfy $a_{n1} < a_{n2}$, $b_{n1} < b_{n2}$ and $c_{n1} < c_{n2}$ and this is shown as follows:

$$\begin{aligned}
& \mu(x) = \text{itritype2}(x, [a_{n1}, b_{n1}, c_{n1}, a_{n2}, b_{n2}, c_{n2}]) \\
& \text{where } a_{n1} < a_{n2}, b_{n1} < b_{n2}, c_{n1} < c_{n2}, \\
& \mu_1(x) = \max\left(\min\left(\frac{x - a_1}{b_1 - a_1}, 1, \frac{c_1 - x}{c_1 - b_1}\right), 0\right) \\
& \mu_2(x) = \max\left(\min\left(\frac{x - a_2}{b_2 - a_2}, 1, \frac{c_2 - x}{c_2 - b_2}\right), 0\right) \tag{4.14} \\
& \bar{\mu}(x) = \begin{cases} \max(\mu_1(x), \mu_2(x)) & \forall x(b_{n1}, b_{n2}), \\ 1 & \forall x \in (b_{n1}, b_{n2}) \end{cases} \\
& \underline{\mu}(x) = \min(\mu_1(x), \mu_2(x))
\end{aligned}$$

In Figures 4.64 and 4.65 we can find the graphical description of interval type-2 triangular and trapezoidal membership functions, respectively.

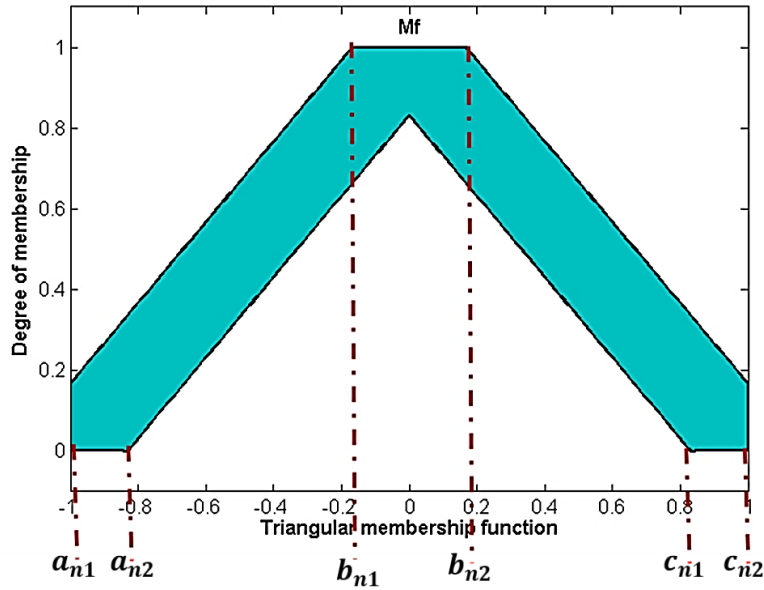


Fig. 4.52 Interval type-2 triangular membership function

The representation of the interval type-2 trapezoidal membership functions (itrapatype2) of the fuzzy controller consists of 8 parameters a_{n1} , b_{n1} , c_{n1} , d_{n1} , a_{n2} , b_{n2} , c_{n2} and d_{n2} where $a_{n1} < a_{n2}$, $b_{n1} < b_{n2}$, $c_{n1} < c_{n2}$ and $d_{n1} < d_{n2}$ and this is shown as follows:

$$\begin{aligned}
\tilde{\mu}(x) &= [\underline{\mu}(x), \bar{\mu}(x)] \\
&= \text{itrapatype2}(x, [a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2, \alpha]) \\
&\text{where } a_1 < a_2, b_1 < b_2, c_1 < c_2, d_1 < d_2 \\
\mu_1(x) &= \max\left(\min\left(\frac{x - a_1}{b_1 - a_1}, 1, \frac{d_1 - x}{d_1 - c_1}\right), 0\right) \\
\mu_2(x) &= \max\left(\min\left(\frac{x - a_2}{b_2 - a_2}, 1, \frac{d_2 - x}{d_2 - c_2}\right), 0\right) \\
\bar{\mu}(x) &= \begin{cases} \max(\mu_1(x), \mu_2(x)) & \forall x \notin (b_1, c_2) \\ 1 & \forall x \in (b_1, c_2) \end{cases} \\
\underline{\mu}(x) &= \min(\alpha, \min(\mu_1(x), \mu_2(x)))
\end{aligned} \tag{4.15}$$

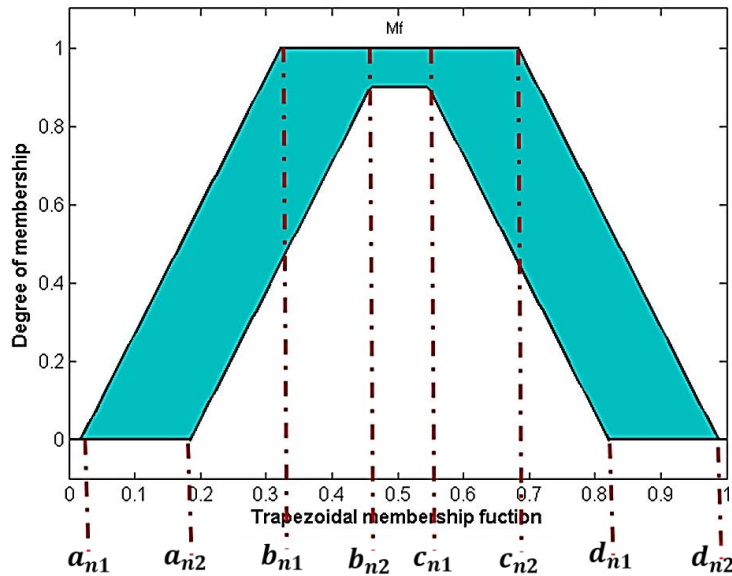


Fig. 4.53 Interval type-2 trapezoidal membership function

To test the proposed methodology, we consider the case of controlling an autonomous mobile robot. For this particular case, the main function of the fuzzy controller is to provide control of the motors to minimize the error in an established trajectory. The robot design for the case study can be found in session 4.1.

The fuzzy controller of the mobile robot is formed by two input variables labeled as the errors of the linear and angular velocities, granulated into three membership functions, two trapezoidal for the edges and one triangular for the center labeled as negative, zero and positive respectively. The output variables are called *torque1* and *torque2*, which are granulated into three triangular membership functions labeled: negative, zero and positive. Below we have a graphic description of the inputs and outputs that form the interval type-2 fuzzy controller of the mobile autonomous robot that, as mentioned above, which are granulated with triangular and trapezoidal membership functions. Figures 4.66 to 4.69 illustrate the corresponding type-2 membership functions.

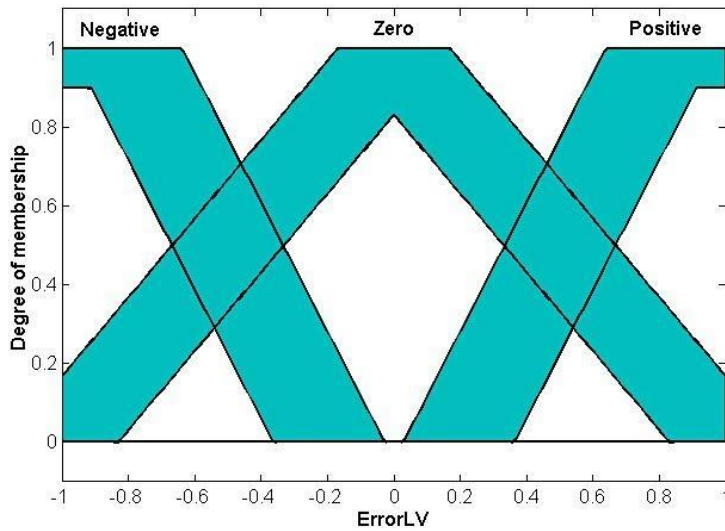


Fig. 4.54 Membership functions of the linear velocity error

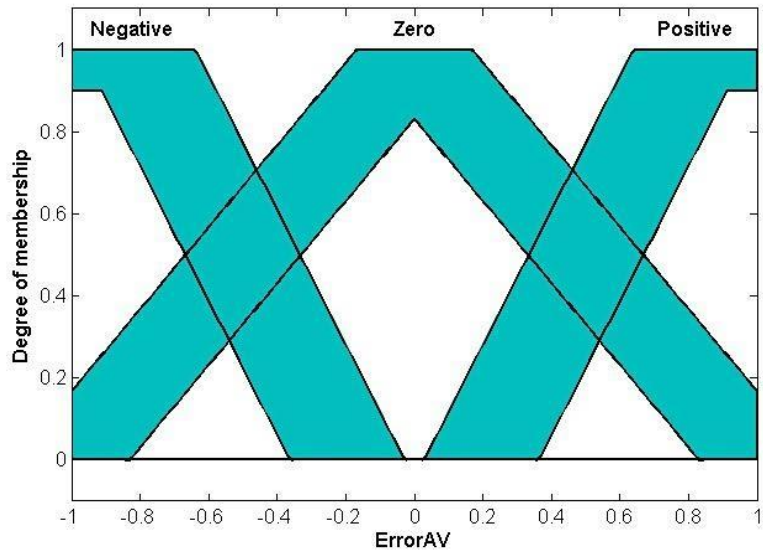


Fig. 4.55 Membership functions of the angular velocity error

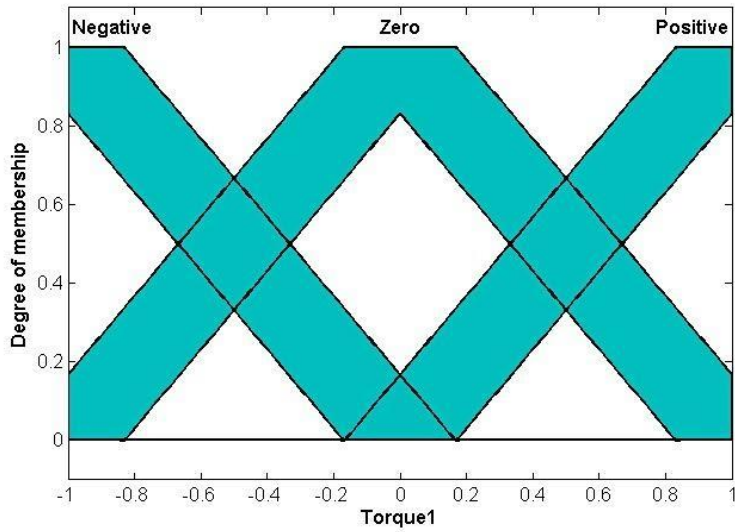


Fig. 4.56 Membership functions of Torque 1

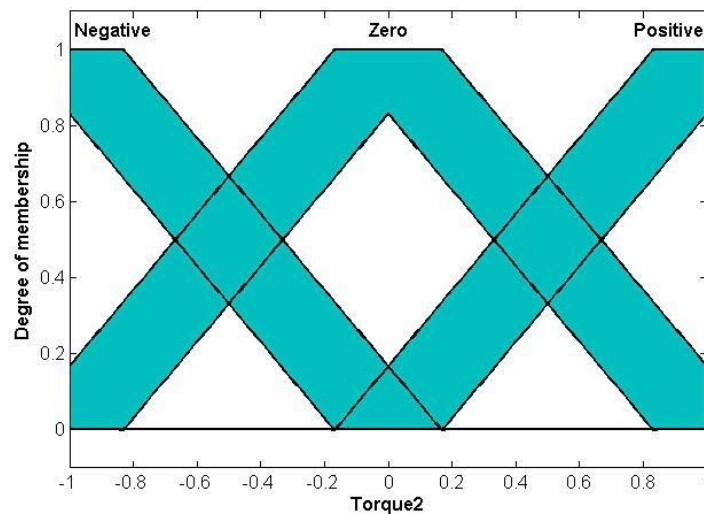


Fig. 4.57 Membership functions of the Torque 2

The design of the fuzzy rules of the autonomous mobile robot controller enables controlling the movement of the robot following the established trajectory, so that the movement is uniform and continuous. The fuzzy rules are listed below:

1. If (ErrorLV is Negatiive) and (ErrorAV is Negative) then (Torque1 is Negative) and (Torque2 is Negative)
2. If (ErrorLV is Negatiive) and (ErrorAV is Zero) then (Torque1 is Negative) and (Torque2 is Zero)
3. If (ErrorLV is Negatiive) and (ErrorAV is Positive) then (Torque1 is Negative) and (Torque2 is Positive)
4. If (ErrorLV is Zero) and (ErrorAV is Negative) then (Torque1 is Zero) and (Torque2 is Negative)
5. If (ErrorLV is Zero) and (ErrorAV is Zero) then (Torque1 is Zero) and (Torque2 is Zero)
6. If (ErrorLV is Zero) and (ErrorAV is Positive) then (Torque1 is Zero) and (Torque2 is Positive)

7. If (ErrorLV is Positive) and (ErrorAV is Negative) then (Torque1 is Positive) and (Torque2 is Negative)
8. If (ErrorLV is Positive) and (ErrorAV is Zero) then (Torque1 is Positive) and (Torque2 is Zero)
9. If (ErrorLV is Positive) and (ErrorAV is Positive) then (Torque1 is Positive) and (Torque2 is Positive)

These fuzzy rules are basically deciding the particular actions on the left and right wheels according to the position of the robot with respect to the desired trajectory.

4.6.2 Experimentation and results of the optimization

In this section, the results obtained from the fuzzy controller optimization of the autonomous mobile robot are presented. The methodology consists on using a metaheuristic algorithm to generate a vector of the necessary parameters to form the membership functions of the interval type-2 fuzzy controller that is optimized. For this specific case, the metaheuristics are the galactic swarm optimization and firefly algorithm and their variants with dynamic adaptation of parameters using type-1 and interval type-2 fuzzy systems. Table 4.70 shows the parameters used in galactic swarm optimization and firefly algorithm to perform the optimization of the fuzzy controller.

Table 4.51 Parameters of the GSO and FA algorithms

Parameter	GSO	Parameter	FA
Population	10	Population	50
Subpopulation	5	Iteration	750
Iteration 1	5	-	-
Iteration 2	10		-
C ₁ and C ₂	2.5		
C ₃ and C ₄	dynamic		

To measure the performance of the algorithms for the optimization of the fuzzy controller, the mean squared error (MSE) is used. Its equation is described below:

$$MSE = \frac{1}{n} \sum_{i=1}^n (X_i - Y_i)^2 \quad (4.18)$$

where:

X_i Is the reference value at time i .

Y_i Is the value produced by the system at time i .

n Is the number of samples considered in the test.

Table 4.71 shows the performed experiments to obtain the best optimized fuzzy system with the FA, where it can be observed that the best error found is of 9.64×10^{-02} and an average of $3.20 \times 10^{+00}$ was obtained.

Table 4.52 FA Results

Experiment	MSE	Experiment	MSE
1	2.52×10^{-01}	16	$7.06 \times 10^{+00}$
2	3.28×10^{-01}	17	$7.08 \times 10^{+00}$
3	5.72×10^{-01}	18	$9.49 \times 10^{+00}$
4	$7.06 \times 10^{+00}$	19	$9.75 \times 10^{+00}$
5	$4.42 \times 10^{+00}$	20	$1.01 \times 10^{+00}$
6	$9.49 \times 10^{+00}$	21	$1.26 \times 10^{+00}$
7	9.64×10^{-02}	22	$2.68 \times 10^{+00}$
8	2.52×10^{-01}	23	$3.13 \times 10^{+00}$
9	3.27×10^{-01}	24	$3.15 \times 10^{+00}$
10	5.25×10^{-01}	25	$1.24 \times 10^{+00}$
11	5.72×10^{-01}	26	$7.35 \times 10^{+00}$
12	$1.13 \times 10^{+00}$	27	$1.21 \times 10^{+00}$
13	$3.84 \times 10^{+00}$	28	$1.34 \times 10^{+00}$

14	$4.42 \times 10^{+00}$	29	$1.16 \times 10^{+00}$
15	$4.43 \times 10^{+00}$	30	$1.29 \times 10^{+00}$

Table 4.72 shows the performed experiments to obtain the best optimized fuzzy system with the fuzzy firefly algorithm, where it can be observed that the best error found is of 9.93×10^{-01} and an average of $1.67 \times 10^{+00}$ was obtained.

Table 4.53 Results for the Fuzzy FA

Experiment	MSE	Experiment	MSE
1	$1.67 \times 10^{+00}$	16	6.63×10^{-01}
2	$1.76 \times 10^{+00}$	17	8.83×10^{-01}
3	3.24×10^{-01}	18	3.06×10^{-01}
4	7.45×10^{-01}	19	8.82×10^{-01}
5	9.70×10^{-01}	20	3.44×10^{-01}
6	3.06×10^{-01}	21	9.98×10^{-01}
7	$1.37 \times 10^{+00}$	22	3.22×10^{-01}
8	9.80×10^{-01}	23	3.31×10^{-01}
9	$1.87 \times 10^{+00}$	24	4.43×10^{-01}
10	$1.89 \times 10^{+00}$	25	8.34×10^{-01}
11	3.64×10^{-01}	26	9.22×10^{-01}
12	3.53×10^{-01}	27	7.73×10^{-01}
13	3.06×10^{-01}	28	3.46×10^{-01}
14	8.79×10^{-01}	29	9.93×10^{-01}
15	3.06×10^{-01}	30	3.06×10^{-01}

Table 4.73 shows the performed experiments to obtain the best optimized fuzzy system with the interval Type-2 Fuzzy firefly algorithm, where it can be observed that the best error found is of 9.64×10^{-02} and an average of $1.06 \times 10^{+00}$ is obtained.

Table 4.54 Results for the Interval Type-2 Fuzzy FA

Experiment	MSE	Experiment	MSE
1	$1.27x10^{+00}$	16	$1.06x10^{+00}$
2	$1.90x10^{+00}$	17	$3.28x10^{-01}$
3	$5.72x10^{-01}$	18	$1.06x10^{+00}$
4	$1.60x10^{+00}$	19	$8.75x10^{-01}$
5	$1.27x10^{+00}$	20	$7.35x10^{-01}$
6	$4.04x10^{-01}$	21	$9.12x10^{-01}$
7	$1.14x10^{+00}$	22	$7.47x10^{-01}$
8	$9.64x10^{+02}$	23	$9.34x10^{-01}$
9	$1.06x10^{+00}$	24	$8.82x10^{-01}$
10	$1.09x10^{+00}$	25	$7.45x10^{-01}$
11	$1.06x10^{+00}$	26	$1.12x10^{-01}$
12	$2.53x10^{-01}$	27	$8.76x10^{-01}$
13	$1.06x10^{+00}$	28	$9.98x10^{-01}$
14	$9.70x10^{-01}$	29	$7.45x10^{-01}$
15	$5.25x10^{-01}$	30	$3.38x10^{-01}$

Table 4.74 shows the performed experiments to obtain the best optimized fuzzy system with the GSO algorithm, where it can be observed that the best error found is of 5.05×10^{-03} and an average of $1.17 \times 10^{+00}$ was obtained.

Table 4.55 Results for Galactic Swarm Optimization (GSO)

Experiment	MSE	Experiment	MSE
1	$8.65x10^{-01}$	16	$3.41x10^{-01}$
2	$7.74x10^{-02}$	17	$7.14x10^{-02}$
3	$2.32x10^{-01}$	18	$7.40x10^{-01}$

4	$1.20x10^{-01}$	19	$1.44x10^{+00}$
5	$2.23x10^{-01}$	20	$9.20x10^{-01}$
6	$1.96x10^{+00}$	21	$1.19x10^{+00}$
7	$1.63x10^{-02}$	22	$7.03x10^{+00}$
8	$5.93x10^{-03}$	23	$4.71x10^{+00}$
9	$4.67x10^{-01}$	24	$5.58x10^{+00}$
10	$2.88x10^{-01}$	25	$8.64x10^{-02}$
11	$5.05x10^{-03}$	26	$8.46x10^{-01}$
12	$6.97x10^{-01}$	27	$3.54x10^{+00}$
13	$2.91x10^{-02}$	28	$3.52x10^{-01}$
14	$6.65x10^{-03}$	29	$3.89x10^{-01}$
15	$3.35x10^{-01}$	30	$2.58x10^{+00}$

Table 4. 75 shows the performed experiments to obtain the best optimized fuzzy system with the FGSO algorithm, where it can be observed that the best error found is of 5.78×10^{-06} and an average of 8.84×10^{-01} was achieved.

Table 4.56 Results for Fuzzy Galactic Swarm Optimization (FGSO1)

Experiment	MSE	Experiment	MSE
1	$4.57x10^{-01}$	16	$2.63x10^{-05}$
2	$1.60x10^{-01}$	17	$3.44x10^{-01}$
3	$1.68x10^{-01}$	18	$3.90x10^{-02}$
4	$1.40x10^{-01}$	19	$5.84x10^{-01}$
5	$5.78x10^{-06}$	20	$4.94x10^{-01}$
6	$3.63x10^{-02}$	21	$6.69x10^{-01}$

7	$1.59x10^{-01}$	22	$5.19x10^{-01}$
8	$6.08x10^{-01}$	23	$5.46x10^{-02}$
9	$2.33x10^{-01}$	24	$2.23x10^{-02}$
10	$3.08x10^{-01}$	25	$2.43x10^{-02}$
11	$1.45x10^{-01}$	26	$1.19x10^{-01}$
12	$1.73x10^{-01}$	27	$8.84x10^{-01}$
13	$1.25x10^{-01}$	28	$8.41x10^{-01}$
14	$3.74x10^{-01}$	29	$1.61x10^{-01}$
15	$8.11x10^{-02}$	30	$1.82x10^{-01}$

Table 4.76 shows the performed experiments to obtain the best optimized the interval type-2 fuzzy system with GSO algorithm, where it can be observed that the best error found is of 1.39×10^{-05} and an average of 8.35×10^{-01} was achieved.

Table 4.57 Results for Interval Type-2 Fuzzy Galactic Swarm Optimization (FGSO IT2)

Experiment	MSE	Experiment	MSE
1	$7.87x10^{-01}$	16	$2.67x10^{-01}$
2	$3.53x10^{-01}$	17	$4.16x10^{-01}$
3	$8.50x10^{-03}$	18	$8.35x10^{-01}$
4	$3.90x10^{-03}$	19	$3.80x10^{-01}$
5	$6.28x10^{-01}$	20	$6.60x10^{-01}$
6	$1.05x10^{-02}$	21	$1.39x10^{-05}$

7	$8.29x10^{-02}$	22	$7.40x10^{-01}$
8	$1.06x10^{-01}$	23	$1.19x10^{-02}$
9	$6.83x10^{-01}$	24	$1.38x10^{-01}$
10	$8.18x10^{-01}$	25	$1.38x10^{-01}$
11	$1.06x10^{-01}$	26	$3.40x10^{-01}$
12	$3.33x10^{-01}$	27	$2.29x10^{-01}$
13	$5.84x10^{-01}$	28	$3.13x10^{-01}$
14	$6.74x10^{-01}$	29	$1.25x10^{-01}$
15	$1.14x10^{-02}$	30	$1.58x10^{-01}$

Table 4.77 shows the best, the worst, the average and standard deviations after 30 executions with the FA and its variants after having performed the fuzzy controller optimization.

Table 4.58 Best, worst and Average for the FA and its variants.

	FA	FA_T1	FA_T2
Best	$9.64x10^{-02}$	$9.93x10^{-01}$	$9.64x10^{-02}$
Worst	$9.75x10^{+00}$	$1.67x10^{+00}$	$1.06x10^{+00}$
Average	$3.20x10^{+00}$	$7.81x10^{-01}$	$8.54x10^{-01}$
Standard Deviation	$3.14x10^{+00}$	$5.03x10^{-01}$	$4.09x10^{-01}$

Table 4.78 shows the best, the worst, the average and standard deviations after 30 executions with the GSO algorithm and its variants after having performed the fuzzy controller optimization.

Table 4.59 Best, worst and Average for the GSO algorithm and its variants

	GSO	FGSO1	FGSO IT2
Best	5.05×10^{-03}	5.78×10^{-06}	1.39×10^{-05}
Worst	$7.04 \times 10^{+00}$	8.84×10^{-01}	8.35×10^{-01}
Average	$1.10 \times 10^{+00}$	2.70×10^{-01}	3.34×10^{-01}
Standard Deviation	$1.77 \times 10^{+00}$	2.48×10^{-01}	2.81×10^{-01}

Finally, the results obtained with the methods are presented in Figures 4.69, 4.70, 4.71 and 4.72, respectively, where one can note the differences between the desired trajectory and the real one generated by the autonomous mobile robot with the type-2 fuzzy logic controller.

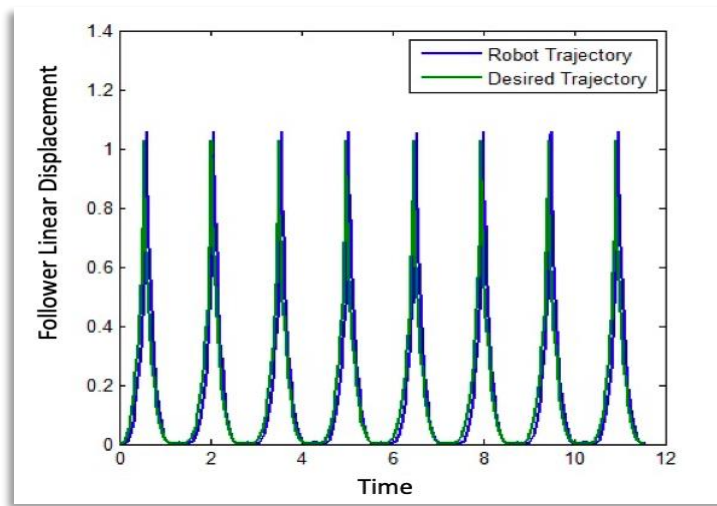


Fig. 4.58 The best result (9.64×10^{-02}) obtained with Firefly Algorithm (FA) and its variants

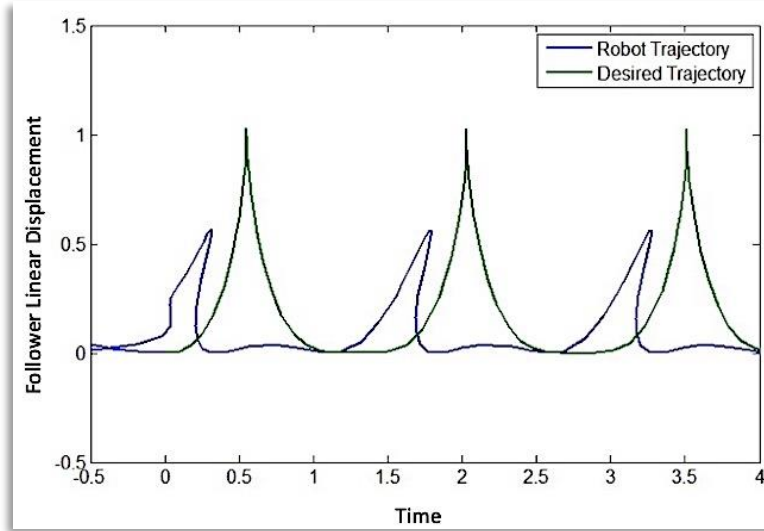


Fig. 4.59 The worst result ($9.75 \times 10^{+00}$) obtained with Firefly Algorithm (FA) and its variants

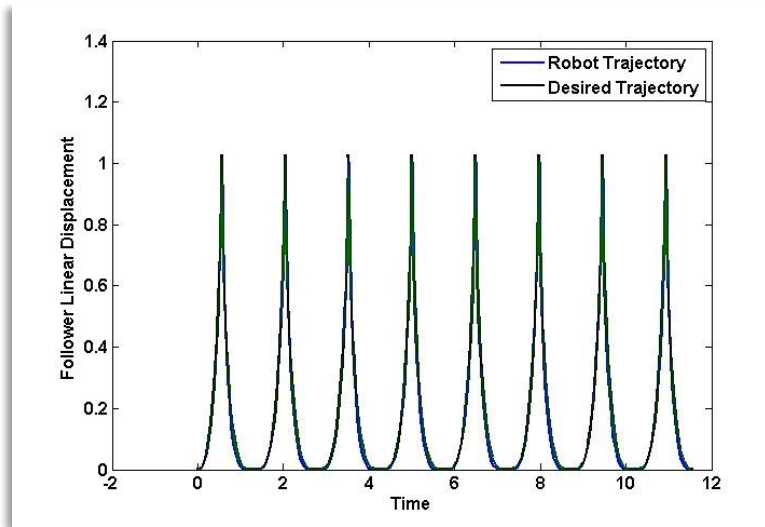


Fig. 4.60 The best result (5.78×10^{-06}) obtain with the GSO algorithm and its variants

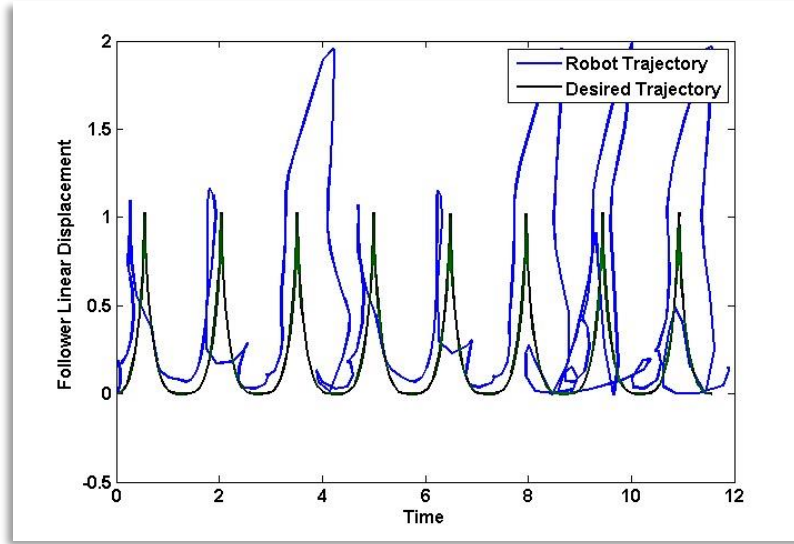


Fig. 4.61 The worst result ($7.04 \times 10^{+00}$) obtain with the GSO algorithm and its variants

To determine which of the algorithms used gives us a better result, 30 experiments are carried out, where the parameters used by each one of them are varied, to determine with which of these variants the algorithm efficiently solves the problem provided.

$$iteration = \frac{Current\ iteration}{Total\ nummber\ iterations} \quad (4.19)$$

4.6.3 Statistical test

In order to validate the performance of the metaheuristic algorithms used to test the proposed optimization methodology of membership functions parameters of the mobile robot fuzzy controller, a statistical comparison is made to find evidence that the algorithms have performed well in the case study of the autonomous mobile robot plant. Table 4.79 shows the representation of the values used in the statistical test, to determine which method produced better results in the optimization of membership function parameters, generating a data vector, which helps to improve the performance of the fuzzy controller for the autonomous robot.

Table 4.60 Statistical test parameters [61], [62]

Parameter	Value
H_0	$\mu_1 \geq \mu_2$
H_a	$\mu_1 < \mu_2$ (Claim)
Level of significance	95%
A	0.05
Critical Value	-1.645

In Table 4.79 the null hypothesis (H_0) states that: the average MSE error for the GSO algorithm and its variants (μ_1), is greater than or equal to the average MSE error for FA and its variants (μ_2). The alternative hypothesis (H_a) states that the average MSE error for the GSO algorithm and its variants (μ_1) is less than the average MSE error for FA and its variants (μ_2).

Table 4.80 shows the averages, standard deviations and values of z for each of the comparisons made, where the GSO algorithm and its variants manage to obtain significant evidence against the FA and its variants. As a consequence, H_0 is rejected and H_a is accepted with a 95% of the level of significance and a rejection zone for values lower than -1.645.

Table 4.61 Results of the Z-test for the GSO algorithm and the FA

GSO		FA		Z value
Average	Std	Average	Std	
$1.10 \times 10^{+00}$	$1.1 \times 10^{+00}$	3.20E + 00	$3.14 \times 10^{+00}$	$-4.85 \times 10^{+00}$
FGSO		FA_T1		
2.70×10^{-01}	2.48×10^{-01}	7.81×10^{-01}	5.03×10^{-01}	$-4.87 \times 10^{+01}$
FGSO IT2		FA_T2		
3.34×10^{-01}	2.8×10^{-01}	8.54×10^{-01}	4.9×10^{-01}	$-6.34 \times 10^{+01}$

In this paper we optimized a type-2 fuzzy controller which controls autonomous robot navigation, and then the results obtained in the experimentation are described. This

experimentation is made with two optimization methods for the type-2 fuzzy controller to make a comparison of the methods and thus, discover which one is the best.

The Friedman test is a non-parametric test to compare multiple algorithms to find the best algorithm. First, if we compare the original GSO and FA methods, we find that there is a significant advantage of GSO with a p value of 0.01 (see Table 4.81). Also, if we compare FGSO and FA_T1 we find significant advantage of FGSO with a p value of 0.00006 (see Table 4.82). Finally, we can compare FGSO IT2 versus FA_T2 and we find significant advantage of FGSO IT2 with a p value of 0.00006 (see Table 4.83).

Table 4.62 Friedman test in comparing GSO versus FA

GSO	vs	FA
Test Statistic		p- value
6.53333		0.01059
$Q = \frac{12n}{k(K+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{3} \right]$		
$Q = 0.06666666 * 4148.0-270$		
$Q = 6.53333$		

Table 4.63 Friedman test in comparing FGSO versus FA_T1

FGSO	vs	FA_T1
Test Statistic		p- value
16.13333		0.00006
$Q = \frac{12n}{k(K+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{3} \right]$		
$Q = 0.06666666 * 4292.0-270$		
$Q = 16.13333$		

Table 4.64 Friedman test in comparing FGSO IT2 versus FA_T2

FGSO IT2	vs	FA_T2
Test Statistic		p- value
16.13333		0.00006
$Q = \frac{12n}{k(K+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{3} \right]$		
$Q = 0.06666666 * 4148.0 - 270$		
$Q = 16.13333$		

In summary we can state that there is sufficient statistical evidence to say that galactic swarm optimization outperforms the firefly algorithm.

In [63] where a Comparative Study of Type-2 Fuzzy Particle Swarm, Bee Colony and Bat Algorithms in Optimization of Fuzzy Controllers is presented. This study makes a comparison between three very promising methods in optimization of fuzzy controllers, where their results for each of the algorithms show best, worst, average, standard deviation, having these references we can compare them with the results obtained in this proposal, where we can discuss which method has achieved better performance in optimization. In Table 4.84 the results of Bee Colony Optimization are illustrated versus results of Table 4.51 that represents Firefly Algorithm (FA), and it is observed that the best value found by the original Bee Colony Optimization (BCO) algorithm, the best with type-1 fuzzy logic (BCO_T1), type-2 fuzzy logic (BCO_T2), but on average and standard deviation the FA showed that performance is better in optimizing membership functions.

Table 4.65 Results of performance the BCO

MSE [38]	BCO	BCO_T1	BCO_T2
Best	8.84x10⁻⁰³	1.50x10⁻⁰³	2.80x10⁻⁰³
Worst	106.81x10 ⁺⁰⁰	61.91x10 ⁺⁰⁰	65.46x10 ⁺⁰⁰

Average	14.61x10 ⁺⁰⁰	9.83x10 ⁺⁰⁰	9.10x10 ⁺⁰⁰
Standard Deviation	23.34x10 ⁺⁰⁰	14.69x10 ⁺⁰⁰	16.67x10 ⁺⁰⁰

Regarding Table 4.85 that is representing Particle Swarm Optimization (PSO Particle Swarm Optimization with type-1 fuzzy logic (PSO_T1), Particle Swarm Optimization with type-2 fuzzy logic (PSO_T2), it is observed that the results were better for PSO than for the FA in Table 4.51, but it must be taken into account that the PSO with inertia weight parameters, position update that helps improve the movement of the particles to find the best overall, and the traditional FA only uses a next position and a Gaussian distribution.

Table 4.66 MSE results of PSO

MSE [38]	PSO	PSO_T1	PSO_T2
Best	1.39x10 ⁻⁰¹	4.41x10 ⁻⁰³	1.90x10⁻⁰⁴
Worst	6.67x10 ⁺⁰⁰	1.51x10 ⁺⁰⁰	3.3x10 ⁻⁰¹
Average	2.41x10 ⁺⁰⁰	3.04x10 ⁻⁰¹	3.22x10 ⁻⁰²
Standard Deviation	2.97x10 ⁺⁰⁰	3.19x10 ⁻⁰¹	7.11x10 ⁻⁰²

In Table 4.86 the Bat Algorithm shows results that are inferior in its optimization performance as shown in Table 4.86 versus the FA of Table 4.84.

Table 4.67 Results of Bat Algorithm

MSE [38]	BA	BA_T1	BA_T2
Best	4x10⁻⁰³	1.47x10 ⁻⁰²	1.64x10 ⁺⁰⁰
Worst	9.31x10 ⁺⁰⁰	97.98x10 ⁺⁰⁰	34.31x10 ⁺⁰⁰
Average	4.02x10 ⁺⁰⁰	36.80x10 ⁺⁰⁰	20.47x10 ⁺⁰⁰
Standard Deviation	2.90x10 ⁺⁰⁰	34.57x10 ⁺⁰⁰	12.25x10 ⁺⁰⁰

Table 4.85 shows the results obtained with GSO, which are better when compared to Tables 4.86 and 4.87. As it is observed, in general the proposed algorithms turn out to have better performance than those found in the literature.

Finally, we show in Table 4.87 a summary of all the above mentioned methods, and we can note that the best fuzzy controller (marked in bold) is found by FGSO IT2 with an error of 1.39×10^{-05} . We can also notice that PSO_T2 achieves the second best fuzzy controller with an error of 1.90×10^{-04} . From the optimal fuzzy control design point of view this is very important because the goal is finding the best possible controller. However, statistically speaking on average there is not a significant difference between the type-2 fuzzy GSO variant and the type-2 fuzzy PSO algorithm.

Table 4.68 Summary of results among the proposed methods and other algorithms in the literature

MSE	Best	Worst	Average	Standard Deviation
FA	9.64×10^{-02}	$9.75 \times 10^{+00}$	$3.20 \times 10^{+00}$	$3.14 \times 10^{+00}$
GSO	5.05×10^{-03}	$7.04 \times 10^{+00}$	$1.67 \times 10^{+00}$	8.84×10^{-01}
BA	4×10^{-03}	$9.31 \times 10^{+00}$	$4.02 \times 10^{+00}$	$2.90 \times 10^{+00}$
BCO	8.84×10^{-03}	$106.81 \times 10^{+00}$	$14.61 \times 10^{+00}$	$23.34 \times 10^{+00}$
PSO	1.39×10^{-01}	$6.67 \times 10^{+00}$	$2.41 \times 10^{+00}$	$2.97 \times 10^{+00}$
FA_T1	9.93×10^{-01}	$1.67 \times 10^{+00}$	7.81×10^{-01}	5.03×10^{-01}
FGSO1	$3.14 \times 10^{+00}$	$1.77 \times 10^{+00}$	5.03×10^{-01}	2.48×10^{-01}
BA_T1	1.47×10^{-02}	$97.98 \times 10^{+00}$	$36.80 \times 10^{+00}$	$34.57 \times 10^{+00}$
BCO_T1	1.50×10^{-03}	$61.91 \times 10^{+00}$	$9.83 \times 10^{+00}$	$14.69 \times 10^{+00}$

PSO_T1	$4.41x10^{-03}$	$1.51x10^{+00}$	$3.04x10^{-01}$	$3.19x10^{-01}$
FA_T2	$9.64x10^{-02}$	$1.0x10^{+00}$	$8.54x10^{-01}$	$4.09x10^{-01}$
FGSO IT2	$1.39x10^{-05}$	$8.35x10^{-01}$	$3.34x10^{-01}$	$2.81x10^{-01}$
BA_T2	$1.64x10^{+00}$	$34.31x10^{+00}$	$20.47x10^{+00}$	$12.25x10^{+00}$
BCO_T2	$2.80x10^{-03}$	$65.4x10^{+00}$	$9.10x10^{+00}$	$16.67x10^{+00}$
PSO_T2	$1.90x10^{-04}$	$3.3x10^{-01}$	$3.22x10^{-02}$	$7.1x10^{-02}$

Chapter 5. Conclusions

In this thesis research, a multi-metaheuristic competitive model was proposed for the optimization of fuzzy systems. Different intelligent computing techniques were used for problem optimization.

The model is composed of several stages, where each algorithm adapts to solve the problem. Its input is the optimization problem to be solved, then each method takes it as an objective function to be able to carry out the exploitation and exploration processes, once the solution is found, it is compared with the stop criterion metric, where it is established if This problem is optimized or not.

The algorithms used for the experimentation were 4 which can be changed depending on the problem. In the experiments, the methods that gave the best results the stochastic fractal search (SFS), then Drone Square optimization (DSO), the next Wind Drive Optimization (WDO) and the one that showed the least efficiency in the results obtained was the Firefly Algorithm (FA). Each of these methods were used according to their optimization performance for each of the cases.

As a conclusion, the smart computing techniques used for fuzzy driver membership function optimization proved to be efficient, and necessary optimizations can be made every day to improve the performance quality of fuzzy drivers.

As future work we can continue with the current model and apply it to type 2 fuzzy controllers, in this way we can see if the methods that had good results with type 1 fuzzy logic also generate good results with type 2 fuzzy logic.

In addition, it can be used in optimization for other optimization problems, such as optimizing algorithms, artificial neural networks.

References

- [1] J. G. Carbonell, R. S. Michalski, and T. M. Mitchell, An Overview of Machine Learning, in *Machine Learning*, Springer Berlin Heidelberg, 1983, pp. 3–23.
- [2] J. E. Laird, C. Lebiere, and P. S. Rosenbloom, A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics, *AI Mag.*, vol. 38, no. 4, pp. 13–26, Dec. 2017.
- [3] J. H. Fetzer, *What is Artificial Intelligence?*, Springer, Dordrecht, 1990, pp. 3–27.
- [4] I. Miramontes, J. C. Guzman, P. Melin, and G. Prado-Arechiga, Optimal design of interval type-2 fuzzy heart rate level classification systems using the bird swarm algorithm, *Algorithms*, vol. 11, no. 12, p. 206, Dec. 2018.
- [5] J. Guzmán, I. Miramontes, P. Melin, and G. Prado-Arechiga, Optimal Genetic Design of Type-1 and Interval Type-2 Fuzzy Systems for Blood Pressure Level Classification, *Axioms*, vol. 8, no. 1, p. 8, Jan. 2019.
- [6] I. Miramontes, P. Melin, and G. Prado-Arechiga, Comparative Study of Bio-inspired Algorithms Applied in the Optimization of Fuzzy Systems, in *Studies in Computational Intelligence*, vol. 827, Springer, 2020, pp. 219–231.
- [7] M. Dorigo and T. Stützle, *The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances*, Springer, Boston, MA, 2003, pp. 250–285.
- [8] A. H. Gandomi, S. Talatahari, X. S. Yang, and S. Deb, Design optimization of truss structures using cuckoo search algorithm, *Struct. Des. Tall Spec. Build.*, vol. 22, no. 17, pp. 1330–1349, Dec. 2013.
- [9] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, A survey on new generation metaheuristic algorithms, *Comput. Ind. Eng.*, vol. 137, p. 106040, Nov. 2019.
- [10] M. Lagunes, O. Castillo, F. Valdez, and J. Soria, Review of Hybrid Combinations of Metaheuristics for Problem Solving Optimization., 2021, p. 221-232.
- [11] L. Amézquita, O. Castillo, J. Soria, and P. Cortes-Antonio, Optimization of Membership Function Parameters for Fuzzy Controllers in Cruise Control Problem Using the Multi-verse Optimizer, in *Studies in Computational Intelligence*, vol. 940, Springer Science and Business Media Deutschland GmbH, 2021, pp. 15–40.
- [12] P. Vasant, I. Zelinka and G. W. Weber. *Intelligent Computing & Optimization*. vol. 866, Springer, 2018.
- [13] P. Melin, O. Castillo, E. G. Ramírez, & W. Pedrycz, (Eds.). (2007). *Analysis and Design of Intelligent Systems Using Soft Computing Techniques*. Vol. 41. Springer Science & Business Media, 2007.
- [14] L. A. Zadeh, “Fuzzy sets,” *Inf. Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.
- [15] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning-III,” *Inf. Sci. (Ny)*, vol. 9, no. 1, pp. 43–80, Jan. 1975.
- [16] P. Hájek, “On very true,” *Fuzzy Sets Syst.*, vol. 124, no. 3, pp. 329–333, Dec. 2001.
- [17] L. A. Zadeh, “ON FUZZY ALGORITHMS,” 1996, pp. 127–147.
- [18] L. A. Zadeh, “Fuzzy logic,” *Computer (Long. Beach. Calif.)*, vol. 21, no. 4, pp. 83–93, Apr. 1988.

- [19] J. M. Mendel, "Advances in type-2 fuzzy sets and systems," *Inf. Sci. (Ny)*, vol. 177, no. 1, pp. 84–110, Jan. 2007.
- [20] J. M. Mendel, R. I. John, and F. Liu, "Interval Type-2 Fuzzy Logic Systems Made Simple," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 6, pp. 808–821, Dec. 2006.
- [21] Qilian Liang and J. M. Mendel, "Interval type-2 fuzzy logic systems: theory and design," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 5, pp. 535–550, 2000.
- [22] J. M. Mendel and H. Wu, "Uncertainty bounds and their use in the design of interval type-2 fuzzy logic systems," *IEEE Trans. FUZZY Syst.*, vol. 10, no. 5, 2002.
- [23] M. A. Sanchez, O. Castillo, and J. R. Castro, "An Overview of Granular Computing Using Fuzzy Logic Systems," Springer, Cham, 2017, pp. 19–38.
- [24] Yang, X. S., Bekdaş, G., & Nigdeli, S. M. (Eds.). *Metaheuristics and Optimization in Civil Engineering*, Springer International Publishing, 2016.
- [25] I. Fister, X.-S. Yang, I. Fister, J. Brest, and D. Fister, "A Brief Review of Nature-Inspired Algorithms for Optimization," *Elektroteh. Vestnik/Electrotechnical Rev.*, vol. 80, no. 3, pp. 116–122, Jul. 2013.
- [26] X.-S. Yang, *Nature-inspired metaheuristic algorithms*. Luniver Press, 2010.
- [27] X. S. Yang and X. He, "Firefly algorithm: recent advances and applications," *Int. J. Swarm Intell.*, vol. 1, no. 1, p. 36, 2013.
- [28] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 1, pp. 89–98, Jan. 2013.
- [29] Eberhart and Yuhui Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 1, pp. 81–86.
- [30] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.
- [31] Z. Bayraktar, M. Komurcu, and D. H. Werner, "Wind Driven Optimization (WDO): A novel nature-inspired optimization algorithm and its application to electromagnetics," in *2010 IEEE Antennas and Propagation Society International Symposium*, Jul. 2010, pp. 1–4.
- [32] A. K. Bhandari, V. K. Singh, A. Kumar, and G. K. Singh, "Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy," *Expert Syst. Appl.*, vol. 41, no. 7, pp. 3538–3560.
- [33] V. V. de Melo, "A novel metaheuristic method for solving constrained engineering optimization problems: Drone Squadron Optimization," Aug. 2017.
- [34] Y. Yalcin and Onur Pekcan, "Nuclear Fission-Nuclear Fusion algorithm for global optimization: a modified Big Bang-Big Crunch algorithm," *Neural Comput. Appl.*, 2020, vol. 32, no 7, p. 2751-2783.
- [35] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, vol. 39, no. 3, pp. 459–471, Nov. 2007.
- [36] J. H. . Holland, "Genetic Algorithms understand Genetic Algorithms," *Sci. Am.*, vol. 267, no. 1, pp. 66–73, 1992.
- [37] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.
- [38] B. L. Galvão Costa, C. Luiz Graciola, B. A. Angélico, A. Goedtel, and M. F. Castoldi,

- “Metaheuristics optimization applied to PI controllers tuning of a DTC-SVM drive for three-phase induction motors,” *Appl. Soft Comput.*, vol. 62, pp. 776–788, 2018.
- [39] S. Khalilpourazari and S. Khalilpourazary, “A Robust Stochastic Fractal Search approach for optimization of the surface grinding process,” *Swarm Evol. Comput.*, vol. 38, pp. 173–186, Feb. 2018.
- [40] Salimi, H., “Stochastic fractal search: a powerful metaheuristic algorithm,” *Knowledge-Based Systems*, 2015, vol. 75, p. 1-18.
- [41] F. Valdez, P. Melin, and O. Castillo, “Toolbox for bio-inspired optimization of mathematical functions,” *Comput. Appl. Eng. Educ.*, vol. 22, no. 1, pp. 11–22, Mar. 2014.
- [42] L. Rodríguez, O. Castillo, M. García, and J. Soria, “A new randomness approach based on sine waves to improve performance in metaheuristic algorithms,” *Soft Comput.*, vol. 24, no. 16, pp. 11989–12011, Aug. 2020.
- [43] O. Carvajal, P. Melin, I. Miramontes, and G. Prado-Arechiga, “Optimal design of a general type-2 fuzzy classifier for the pulse level and its hardware implementation,” *Eng. Appl. Artif. Intell.*, vol. 97, p. 104069, Jan. 2021.
- [44] D. Hidalgo, L. Cervantes, O. Castillo, P. Melin, and R. M. Soto, “Fuzzy parameter adaptation in genetic algorithms for the optimization of fuzzy integrators in modular neural networks for multimodal biometry,” *Comput. y Sist.*, vol. 24, no. 3, pp. 1093–1105, Sep. 2020.
- [45] C. Peraza, F. Valdez, and O. Castillo, “Harmony Search with Dynamic Adaptation of Parameters for the Optimization of a Benchmark Set of Functions,” 2020, pp. 97–108.
- [46] P. Ochoa, O. Castillo, and J. Soria, “Optimization of fuzzy controller design using a Differential Evolution algorithm with dynamic parameter adaptation based on Type-1 and Interval Type-2 fuzzy systems,” *Soft Comput.*, vol. 24, no. 1, pp. 193–214, Jan. 2020.
- [47] L. Astudillo, P. Melin, and O. Castillo, “Optimization of a Fuzzy Tracking Controller for an Autonomous Mobile Robot under Perturbed Torques by Means of a Chemical Optimization Paradigm,” Springer, Berlin, Heidelberg, 2013, pp. 3–20.
- [48] P. Melin, L. Astudillo, O. Castillo, F. Valdez, and M. Garcia, “Optimal design of type-2 and type-1 fuzzy tracking controllers for autonomous mobile robots under perturbed torques using a new chemical optimization paradigm,” *Expert Syst. Appl.*, vol. 40, no. 8, pp. 3185–3195, Jun. 2013.
- [49] M. L. Lagunes, O. Castillo, F. Valdez, and J. Soria, “Multi-Metaheuristic Competitive Model for Optimization of Fuzzy Controllers,” *Algorithms*, vol. 12, no. 5, p. 90, Apr. 2019.
- [50] F. Olivas, F. Valdez, O. Castillo, C. I. Gonzalez, G. Martinez, and P. Melin, “Ant colony optimization with dynamic parameter adaptation based on interval type-2 fuzzy logic systems,” *Appl. Soft Comput. J.*, vol. 53, pp. 74–87, Apr. 2017.
- [51] M. L. Lagunes, O. Castillo, and J. Soria, *Methodology for the optimization of a fuzzy controller using a bio-inspired algorithm*, vol. 648. 2018.
- [52] M. L. Lagunes, O. Castillo, F. Valdez, J. Soria, and P. Melin, “A New Approach for Dynamic Stochastic Fractal Search with Fuzzy Logic for Parameter Adaptation,” *Fractal Fract.*, vol. 5, no. 2, p. 33, Apr. 2021.
- [53] S. Khalilpourazari, B. Naderi, and S. Khalilpourazary, “Multi-Objective Stochastic Fractal Search: a powerful algorithm for solving complex multi-objective optimization problems,” *Soft Comput.*, vol. 24, no. 4, pp. 3037–3066, Feb. 2020.
- [54] O. Castillo, P. Melin, J. Kacprzyk, and W. Pedrycz, “Type-2 Fuzzy Logic: Theory and Applications,” in *2007 IEEE International Conference on Granular Computing (GRC*

- 2007), Nov. 2007, pp. 145–145.
- [55] J. Castro, O. Castillo, P. Melin, A. R.-D.-I. Sciences, and undefined 2009, “A hybrid learning algorithm for a class of interval type-2 fuzzy neural networks,” *Information Sciences*, 2009, vol. 179, no 13, p. 2175-2193.
 - [56] Awad, N. H., et al. Evaluation Criteria for the CEC 2017 Special Session and competition on single objective real-parameter numerical optimization. Technology Report, 2016.
 - [57] İ. B. Aydilek and I. B. Aydilek, “A Hybrid Firefly and Particle Swarm Optimization Algorithm for Computationally Expensive Numerical Problems You can find Source Codes of algorithm here: A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems,” *Appl. Soft Comput.*, vol. 66, pp. 232–249, 2018.
 - [58] E. Bernal, M. L. Lagunes, O. Castillo, J. Soria, and F. Valdez, “Optimization of Type-2 Fuzzy Logic Controller Design Using the GSO and FA Algorithms,” *Int. J. Fuzzy Syst.*, vol. 23, no. 1, pp. 42–57, Feb. 2021.
 - [59] F. Valdez, O. Castillo, and P. Melin, “Bio-Inspired Algorithms and Its Applications for Optimization in Fuzzy Clustering,” *Algorithms*, vol. 14, no. 4, p. 122, Apr. 2021.
 - [60] F. Cuevas, O. Castillo, and P. Cortes, “Towards a Control Strategy Based on Type-2 Fuzzy Logic for an Autonomous Mobile Robot,” in *Studies in Computational Intelligence*, vol. 827, Springer, 2020, pp. 301–314.
 - [61] E. Bernal, O. Castillo, J. Soria, F. Valdez, and P. Melin, “A variant to the dynamic adaptation of parameters in galactic swarm optimization using a fuzzy logic augmentation,” in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Jul. 2018, pp. 1–7.
 - [62] M. J. Mahmoodabadi and H. Jahanshahi, “Multi-objective optimized fuzzy-PID controllers for fourth order nonlinear systems,” *Eng. Sci. Technol. an Int. J.*, vol. 19, no. 2, pp. 1084–1098, Jun. 2016.
 - [63] F. Olivas et al., “Comparative Study of Type-2 Fuzzy Particle Swarm, Bee Colony and Bat Algorithms in Optimization of Fuzzy Controllers,” *Algorithms*, vol. 10, no. 3, p. 101, Aug. 2017.

Appendix

In this section, the source code of each of the methods used in the proposed model for the optimization of fuzzy controllers is presented as an annex.

Appendix A. Code developed in MATLAB

First the Firefly algorithm code is presented, then the wind drive optimization, continuing with drone squadron optimization and finally with Stochastic Fractal Search.

Firefly algorithm code in MATLAB

```
% ===== %
% Files of the Matlab programs included in the book:   %
% Xin-She Yang, Nature-Inspired Metaheuristic Algorithms, %
% Second Edition, Luniver Press, (2010). www.luniver.com %
% ===== %
% ----- %
% Firefly Algorithm for constrained optimization using   %
% for the design of a spring (benchmark)             %
% by Xin-She Yang (Cambridge University) Copyright @2009 %
% ----- %
function fa_ndim
% parameters [n N_iteration alpha betamin gamma]
para=[20 500 0.5 0.2 1];
help fa_ndim.m
% Simple bounds/limits for d-dimensional problems
d=15;
Lb=zeros(1,d);
Ub=2*ones(1,d);
% Initial random guess
u0=Lb+(Ub-Lb).*rand(1,d);
[u,fval,NumEval]=ffa_mincon(@cost,u0,Lb,Ub,para);
% Display results
bestsolution=u
bestobjb=fval
total_number_of_function_evaluations=NumEval
%% Put your own cost/objective function here -----%%
%% Cost or Objective function
function z=cost(x)
% Exact solutions should be (1,1,...,1)
z=sum((x-1).^2);
%% End of the part to be modified -----%%
%% -----%%
%% Do not modify the following codes unless you want %%
%% to improve its performance etc           %%
% -----
% ===Start of the Firefly Algorithm Implementation =====
%     Lb = lower bounds/limits
%     Ub = upper bounds/limits
%     para == optional (to control the Firefly algorithm)
```

```

% Outputs: nbest = the best solution found so far
%         fbest = the best objective value
%         NumEval = number of evaluations: n*MaxGeneration
% Optional:
% The alpha can be reduced (as to reduce the randomness)
% -----
% Start FA
function [nbest,fbest,NumEval]...
    =ffa_mincon(fhandle,u0, Lb, Ub, para)
% Check input parameters (otherwise set as default values)
if nargin<5, para=[20 500 0.25 0.20 1]; end
if nargin<4, Ub=[]; end
if nargin<3, Lb=[]; end
if nargin<2,
disp('Usage: FA_mincon(@cost,u0,Lb,Ub,para)');
end
% n=number of fireflies
% MaxGeneration=number of pseudo time steps
% -----
% alpha=0.25;    % Randomness 0--1 (highly random)
% betamn=0.20;   % minimum value of beta
% gamma=1;      % Absorption coefficient
% -----
n=para(1); MaxGeneration=para(2);
alpha=para(3); betamin=para(4); gamma=para(5);
% Total number of function evaluations
NumEval=n*MaxGeneration;
% Check if the upper bound & lower bound are the same size
if length(Lb) ~=length(Ub),
    disp('Simple bounds/limits are improper!');
    return
end
% Calcualte dimension
d=length(u0);
% Initial values of an array
zn=ones(n,1)*10^100;
% -----
% generating the initial locations of n fireflies
[ns,Lightn]=init_ffa(n,d,Lb,Ub,u0);
% Iterations or pseudo time marching
for k=1:MaxGeneration, %%%%%%%%% start iterations
% This line of reducing alpha is optional
    alpha=alpha_new(alpha,MaxGeneration);
% Evaluate new solutions (for all n fireflies)
for i=1:n,
    zn(i)=fhandle(ns(i,:));
    Lightn(i)=zn(i);
end
% Ranking fireflies by their light intensity/objectives
[Lightn,Index]=sort(zn);
ns_tmp=ns;
for i=1:n,
    ns(i,:)=ns_tmp(Index(i,:));
end
%% Find the current best
nso=ns; Lighto=Lightn;
nbest=ns(1,:); Lightbest=Lightn(1);
% For output only
fbest=Lightbest;

```



```

% Move all fireflies to the better locations
[ns]=ffa_move(n,d,ns,Lightn,nso,Lighto,nbest,...
    Lightbest,alpha,betamin,gamma,Lb,Ub);
end %%%%%%%%% end of iterations
% -----
% ---- All the subfunctions are listed here -----
% The initial locations of n fireflies
function [ns,Lightn]=init_ffa(n,d,Lb,Ub,u0)
    % if there are bounds/limits,
    if length(Lb)>0,
        for i=1:n,
            ns(i,:)=Lb+(Ub-Lb).*rand(1,d);
        end
    else
        % generate solutions around the random guess
        for i=1:n,
            ns(i,:)=u0+randn(1,d);
        end
    end
    % initial value before function evaluations
    Lightn=ones(n,1)*10^100;
    % Move all fireflies toward brighter ones
    function [ns]=ffa_move(n,d,ns,Lightn,nso,Lighto,...
        nbest,Lightbest,alpha,betamin,gamma,Lb,Ub)
    % Scaling of the system
    scale=abs(Ub-Lb);
    % Updating fireflies
    for i=1:n,
        % The attractiveness parameter beta=exp(-gamma*r)
        for j=1:n,
            r=sqrt(sum((ns(i,:)-ns(j,:)).^2));
            % Update moves
            if Lightn(i)>Lighto(j), % Brighter and more attractive
                beta0=1; beta=(beta0-betamin)*exp(-gamma*r.^2)+betamin;
                tmpf=alpha.*(rand(1,d)-0.5).*scale;
                ns(i,:)=ns(i,:).*(1-beta)+nso(j,:).*beta+tmpf;
            end
        end % end for j
    end % end for i
    % Check if the updated solutions/locations are within limits
    [ns]=findlimits(n,ns,Lb,Ub);
    % This function is optional, as it is not in the original FA
    % The idea to reduce randomness is to increase the convergence,
    % however, if you reduce randomness too quickly, then premature
    % convergence can occur. So use with care.
    function alpha=alpha_new(alpha,NGen)
    % alpha_n=alpha_0(1-delta)^NGen=10^(-4);
    % alpha_0=0.9
    delta=1-(10^(-4)/0.9)^(1/NGen);
    alpha=(1-delta)*alpha;
    % Make sure the fireflies are within the bounds/limits
    function [ns]=findlimits(n,ns,Lb,Ub)
    for i=1:n,
        % Apply the lower bound
        ns_tmp=ns(i,:);
        l=ns_tmp<Lb;
        ns_tmp(l)=Lb(l);
        % Apply the upper bounds
        J=ns_tmp>Ub;

```

```

ns_tmp(J)=Ub(J);
% Update this new move
ns(i,:)=ns_tmp;
end
%% ===== End of Firefly Algorithm implementation =====

```

Wind Driven Optimization code in MATLAB

```

%-----
% Sample Matlab / Octave Code for the Wind Driven Optimization.
% Optimization of the Sphere Function in the range of [-5, 5].
% by Dr. Zikri Bayraktar - thewdoalgorithm@gmail.com
%
% DISCLAIMER: This code is provided for educational purposes
% only. Use at own your risk!
%-----
%
% Please refer to the following journal article in your research papers:
% Z. Bayraktar, M. Komurcu, J. A. Bossard and D. H. Werner, "The Wind
% Driven Optimization Technique and its Application in Electromagnetics,"
% IEEE Transactions on Antennas and Propagation, Volume 61, Issue 5,
% pages 2745 - 2757, May 2013.
%-----
tic;
clear;
close all;
clc;
format long g;
delete('WDOoutput.txt');
delete('WDOpressure.txt');
delete('WDOposition.txt');
fid=fopen('WDOoutput.txt','a');
%-----
% User defined WDO parameters:
param.popsize = 20;           % population size.
param.npar = 5;              % Dimension of the problem.
param.maxit = 500;          % Maximum number of iterations.
param.RT = 3;                % RT coefficient.
param.g = 0.2;               % gravitational constant.
param.alp = 0.4;             % constants in the update eq.
param.c = 0.4;               % coriolis effect.
maxV = 0.3;                  % maximum allowed speed.
dimMin = -5;                 % Lower dimension boundary.
dimMax = 5;                  % Upper dimension boundary.
%-----
% Initialize WDO population, position and velocity:
% Randomize population in the range of [-1, 1]:
pos = 2*(rand(param.popsize,param.npar)-0.5);
% Randomize velocity:
vel = maxV * 2 * (rand(param.popsize,param.npar)-0.5);

%-----
% Evaluate initial population: (Sphere Function)
for K=1:param.popsize,
    x = (dimMax - dimMin) * ((pos(K,:)+1)./2) + dimMin;

```

```

        pres(K,:) = sum (x.^2);
end
%-----
% Finding best air parcel in the initial population :
[globalpres,indx] = min(pres);
globalpos = pos(indx,:);
minpres(1) = min(pres);           % minimum pressure
%-----
% Rank the air parcels:
[sorted_pres rank_ind] = sort(pres);
% Sort the air parcels:
pos = pos(rank_ind,:);
keepglob(1) = globalpres;
%-----
% Start iterations :
iter = 1; % iteration counter
for ij = 2:param.maxit,
    % Update the velocity:
    for i=1:param.popsize
        % choose random dimensions:
        a = randperm(param.npar);
        % choose velocity based on random dimension:
        velot(i,:) = vel(i,a);
        vel(i,:) = (1-param.alp)*vel(i,:)-(param.g*pos(i,:))+ ...
                    abs(1-1/i)*((globalpos-pos(i,:)).*param.RT)+ ...
                    (param.c*velot(i,:)/i);
    end

    % Check velocity:
    vel = min(vel, maxV);
    vel = max(vel, -maxV);
    % Update air parcel positions:
    pos = pos + vel;
    pos = min(pos, 1.0);
    pos = max(pos, -1.0);
    % Evaluate population: (Pressure)
    for K=1:param.popsize,
        x = (dimMax - dimMin) * ((pos(K,:)+1)./2) + dimMin;
        pres(K,:) = sum (x.^2);
    end
%-----
% Finding best particle in population
[minpres,indx] = min(pres);
minpos = pos(indx,:);           % min location for this iteration
%-----
% Rank the air parcels:
[sorted_pres rank_ind] = sort(pres);
% Sort the air parcels position, velocity and pressure:
pos = pos(rank_ind,:);
vel = vel(rank_ind,:);
pres = sorted_pres;

% Updating the global best:
better = minpres < globalpres;
if better
    globalpres = minpres           % initialize global minimum
    globalpos = minpos;
end
% Keep a record of the progress:

```

```

        keepglob(ij) = globalpres;
        save WDOposition.txt pos -ascii -tabs;
end
    %Save values to the final file.
    pressure = transpose(keepglob);
    save WDOpressure.txt pressure -ascii -tabs;

    % Plot the pressure function progress over iterations:
    semilogy(keepglob, 'k' , 'LineWidth',2)
    title(['Global Best Pressure is " ', num2str(keepglob(1,param.maxit)), ' " .'])
    xlabel('Number of Iterations')
    ylabel('Global Pressure (i.e. fitness) in log scale')
    grid on
    xlim([0, param.maxit])

    %END
%-----

```

Drone Squadron Optimization code in MATLAB

```

% DRONE SQUADRON OPTIMIZATION Algorithm (DSO) toolbox
% Source codes demo version 1.0
%
% Developed in Octave 3.8. Compatible with MATLAB 2011
%
% Author and programmer: Vinicius Veloso de Melo
%
% e-Mail: dr.vmelo@gmail.com

```

```
%          vinius.melo@unifesp.br
%
%   Homepage: http://www.sjc.unifesp.br/docente/vmelo/
%
% Main paper:
% de Melo, V.V. & Banzhaf, W. Neural Comput & Applic (2017). doi:10.1007/s00521-017-
2881-3
% link.springer.com/article/10.1007/s00521-017-2881-3
% _____
```

```
% Input: the best solutions
% Output: the multivariate normal distribution matrix following the distribution of the selected
best solutions
```

```
function mat = CMA (LocalBestPosition, CurrentOFV, N)
```

```
    [sorted, indices] = sort(CurrentOFV);
```

```
    bestidx = indices (1: ceil(length(indices)*0.5));
```

```
    mu = mean(LocalBestPosition (bestidx, :), 1);
```

```
    sigma = cov( LocalBestPosition(bestidx, :));
```

```
    mat = mvnrnd(mu, sigma, N);
```

```
end
```

```
% _____
```

```
% DRONE SQUADRON OPTIMIZATION Algorithm (DSO) toolbox
```

```
% Source codes demo version 1.0
```

```
%
```

```
% Developed in Octave 3.8. Compatible with MATLAB 2011
```

```
%
```

```
% Author and programmer: Vinicius Veloso de Melo
```

```
%
```

```
%   e-Mail: dr.vmelo@gmail.com
```

```
%          vinius.melo@unifesp.br
```

```
%
```

```
%   Homepage: http://www.sjc.unifesp.br/docente/vmelo/
```

```
%
% Main paper:
% de Melo, V.V. & Banzhaf, W. Neural Comput & Applic (2017). doi:10.1007/s00521-017-
2881-3
% link.springer.com/article/10.1007/s00521-017-2881-3
% _____
```

```
% Input: an array
% Output: The objective function value
```

```
function Cost = CostFunction( x )
    dimension = length(x);
    Cost = sum(100*(x(2:dimension)-x(1:dimension-1)).^2).^2 + (1-x(1:dimension-1)).^2);
% Rosenbrock function
end
```

```
%
% function Cost = CostFunction( x )
% dim=size(x,2);
% Cost=sum(x.^2); % Sphere function
% end
%
```

```
% _____
```

```
% DRONE SQUADRON OPTIMIZATION Algorithm (DSO) toolbox
% Source codes demo version 1.0
%
% Developed in Octave 3.8. Compatible with MATLAB 2011
%
% Author and programmer: Vinicius Veloso de Melo
%
% e-Mail: dr.vmelo@gmail.com
%         vinicius.melo@unifesp.br
%
% Homepage: http://www.sjc.unifesp.br/docente/vmelo/
%
% Main paper:
```

% de Melo, V.V. & Banzhaf, W. Neural Comput & Applic (2017). doi:10.1007/s00521-017-2881-3

% link.springer.com/article/10.1007/s00521-017-2881-3

%

% You can simply define your cost in a separate file and load its handle to fobj

% The initial parameters that you need are:

%

% ObjectiveFunction = @YourCostFunction

% Nvars = number of your variables

% LB=[lb1,lb2,...,lbn] where lbn is the lower bound of variable n

% UB=[ub1,ub2,...,ubn] where ubn is the upper bound of variable n

% setup = DSO configuration

% If all the variables have equal lower bound you can just

% define lb and ub as two single number numbers

% To run DSO: [X,Fx, CurveOFV, NFE, StopMessage, HistoryRanks, Firmwares,
LocalBestPosition] = DSO(ObjectiveFunction, Nvars, LB, UB, setup)

%

function [X,Fx, StatsCurvesOFV, NFE, StopMessage, HistoryRanks, Firmwares,
LocalBestPosition] = DSO(ObjectiveFunction, Nvars, LB, UB, setup)

 stderr = 1;

 more off

 global Teams_OFV

 addpath ('./GPOLS')

 %% Check Input

 if nargin < 4

 fprintf(stderr, '\n!!! Call dso(Nvars, LB, UB) or dso(Nvars, LB, UB, setup)\n');

 error('DSO : Not Enough Input Arguments!')

 end

 if ~isequal(Nvars, size(LB, 2), size(UB, 2))

```
error('DSO : Invalid Arguments : isequal(Nvars,size(LB,2),size(UB,2)) should be true ');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Default Parameters
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
C1 = 0.8;
C2 = 0.012;
C3 = 0.72;
p=0.25;
Pacc = 0.1;
ConvThres = 1e-2;
MaxIterations = 10;
maxtreedepth = 10;
Command_Center_Iter = 100;
use_known_perturbation = 1;
N = 30;
N_Teams = 4;
ReportLag = 100;
NFE = 0;
VTR = -Inf;
MaxStagnation = 10;
Toolbox = false;
StopMessage = 'Stopping DSO : Reached Maximum Number of Iterations';
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% Update setup
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```



```

if isfield(setup, 'C1'), C1 = setup.C1 ; end
if isfield(setup, 'C2'), C2 = setup.C2 ; end
if isfield(setup, 'C3'), C3 = setup.C3 ; end
if isfield(setup, 'N'), N = setup.N ; end
if isfield(setup, 'N_Teams'), N_Teams = setup.N_Teams ; end
if isfield(setup, 'MaxIterations'), MaxIterations = setup.MaxIterations ; end
if isfield(setup, 'VTR'), VTR = setup.VTR ; end
if isfield(setup, 'ReportLag'), ReportLag = setup.ReportLag ; end
if isfield(setup, 'MaxStagnation'), MaxStagnation = setup.MaxStagnation ; end
if isfield(setup, 'Command_Center_Iter'), Command_Center_Iter =
setup.Command_Center_Iter ; end
if isfield(setup, 'Pacc'), Pacc = setup.Pacc ; end
if isfield(setup, 'ConvThres'), ConvThres = setup.ConvThres ; end
if isfield(setup, 'Toolbox'), Toolbox = setup.Toolbox; end

```

%%%%%%%%%%
%%%%%%%%%%

Stagnation=0;

```

disp ('');
fprintf('=====\n', ...
'Using the following setup:\n ',...
'C1 = %f\n ',...
'C2 = %f\n ',...
'C3 = %f\n ',...
'Maximum Iterations = %d\n ',...
'Squadron size = %d\n ',...
'Number of teams = %d\n ',...
'Report Lag = %d\n ',...
'=====\n\n', C1, C2, C3, MaxIterations, N,
N_Teams, ReportLag);

```

%%%%%%%% AUXILIARY FUNCTIONS %%%%

```

U0_1 = @(N) repmat(randdraw('unif', [0, 1], N, 1), 1, Nvars);
U05_1 = @(N) repmat(randdraw('unif', [0.5, 1], N, 1), 1, Nvars);
N0_1 = @(N) repmat(randdraw('normal', [0, 1], N, 1), 1, Nvars);

```

```
N05_01 = @(N) repmat(randdraw('normal', [0.5, 0.1], N, 1), 1, Nvars);
N0_01 = @(N) repmat(randdraw('normal', [0, 0.1], N, 1), 1, Nvars);
```

```
Opposition = @(LocalBestPosition, sumIntervals) sumIntervals - LocalBestPosition;
```

```
avg = @(a, b) (a+b)/2;
%pdiv = @(a,b) (abs(b) > 1e-20) .* a./b;
plog = @(a) (a>0) .* log(a);
square = @(a) a .^ 2;
psqrt = @(a) sign(a) .* sqrt(abs(a));
neg = @(a) (-1) * a;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%% GENERATE INITIAL LARGE (N * N_Teams) SAMPLE FROM THE
LANDSCAPE
```

```
%% Initial Position
CurrentPosition = zeros(N*N_Teams,Nvars); % Initial Position
for i = 1:Nvars
    CurrentPosition(:,i) = randdraw('unif', [LB(i),UB(i)],N*N_Teams,1);
end
```

```
%% Evaluate Initial Position
CurrentOFV = zeros(N*N_Teams,1); % OFV Value
for i = 1:N*N_Teams
    CurrentOFV(i) = ObjectiveFunction(CurrentPosition(i,:));
    NFE = NFE + 1;
end
```

```
%% Select the N best solutions
[sorted, indices] = sort(CurrentOFV);
```

```
%% Update Local Best
LocalBestPosition = CurrentPosition( indices(1:N), : ); % Local Best
LocalBestOFV = CurrentOFV ( indices(1:N) );
```

```
pbest = ceil(N*p);
```

```

posPBest = randi(pbest, 1, N); %indices(randi(pbest, 1, N));

CurrentPosition = LocalBestPosition ;
CurrentOFV = LocalBestOFV;

%% Update Global Best
[GlobalBestOFV, index] = min(LocalBestOFV);
GlobalBestPosition = repmat(LocalBestPosition(index,:),N,1); % Global Best

FinalBestOFV = GlobalBestOFV;
FinalBestSolution = GlobalBestPosition(1,:);
FinalBestNFE = NFE;

%% Initial Shift ( Velocity )
Shift = CurrentPosition .* rand(N,Nvars) ;

matInterval = repmat(abs(UB-LB),N,1);
sumIntervals = repmat(abs(UB+LB),N,1);

%% Initialize Teams and Firmwares

Teams_Positions = cell(1, N_Teams); % matrix of solutions
Teams_Positions_Evaluation = Inf * ones(N, N_Teams); % matrix of objective function
values
Teams_Ranking = Teams_Positions_Evaluation; % matrix of objective function values
Teams_OFV = zeros(1, N_Teams);

Firmwares = Teams_Positions;

GPConfig = InitGPConfig(N_Teams, maxtreedepth, use_known_perturbation);

for i = 1:N_Teams
    Firmwares{i} = tree_stringrc( GPConfig.popu.chrom{i}.tree, 1, GPConfig.symbols );
end

fprintf(stderr, 'INITIAL FIRMWARES:');
disp(Firmwares);

%%% MAIN LOOP

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
StatsCurvesOFV = Inf(MaxIterations, 5);
```

```
StatsCurvesOFV(1, :) = [GlobalBestOFV, min(CurrentOFV), mean(CurrentOFV),  
median(CurrentOFV), max(CurrentOFV)];
```

```
%CurveOFV = Inf(1, MaxIterations);
```

```
%CurveOFV(1) = GlobalBestOFV;
```

```
PreviousGlobalBestOFV = Inf;
```

```
rot = (0:1:N-1);          % rotating index array (size NP)
```

```
rotd= (0:1:Nvars-1);     % rotating index array (size D)
```

```
HistoryRanks = Inf(MaxIterations, N_Teams);
```

```
%% Start search process
```

```
for Iter = 2:MaxIterations
```

```
    if abs(FinalBestOFV) <= VTR  
        fprintf(stderr, 'VTR FOUND!!! ');  
        break;  
        continue;  
    end
```

```
    if NFE > setup.MaxEvaluations  
        fprintf(stderr, '!!! NFE > setup.MaxEvaluations !!!\n');  
        break;  
    end
```

```
    ind = randperm(4);      % index pointer array
```

```
    a1 = randperm(N);      % shuffle locations of vectors  
    rt = rem(rot+ind(1),N); % rotate indices by ind(1) positions  
    a2 = a1(rt+1);         % rotate vector locations  
    rt = rem(rot+ind(2),N);  
    a3 = a2(rt+1);
```

```

rt = rem(rot+ind(3),N);
a4 = a3(rt+1);
rt = rem(rot+ind(4),N);
a5 = a4(rt+1);

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MOVE THE TEAMS USING THE Firmwares
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

violations = zeros(1, N_Teams);

```

```

for team = 1:N_Teams

```

```

    %% CALCULATE

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

    %% Update Velocity and Position

```

```

        Trial_Position = eval(Firmwares{team});

```

```

        mui = rand(N, Nvars) < 0.4 + rand*0.5;    % all random numbers < CR are 1, 0
otherwise

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

    %% RECOMBINATION

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

    %% Choose a recombination operator

```

```

    opCrossOver = randi(3);

```

```

    if (opCrossOver == 1) % BINOMIAL crossover

```

```

        mpo = mui < 0.5;    % inverse mask to mui

```

```

        jrand = randi(Nvars);

```

```

New_Team_Position = LocalBestPosition.*mpo + Trial_Position.*mui;

elseif (opCrossOver == 2) % EXPONENTIAL CROSSOVER
    mui=sort(mui');      % transpose, collect 1's in each column
    for i=1:N
        n=floor(rand*Nvars);
        if n > 0
            rtd = rem(rotd+n,Nvars);
            mui(:,i) = mui(rtd+1,i); %rotate column i by n
        end
    end
    mui = mui';      % transpose back
    mpo = mui < 0.5;      % inverse mask to mui

    jrand = randi(Nvars);

    New_Team_Position = LocalBestPosition.*mpo + Trial_Position.*mui;

else % NO CROSSOVER
    New_Team_Position = Trial_Position;
end

%% BOUND CORRECTION

%% Choose a bound correction approach
opBounds = randi(3);

%% Disable bound correction if it is in the setup
if (setup.checkBounds == 0)
    opBounds = -1;
end

if (opBounds == 1)
    for i = 1:Nvars
        indexes = find(New_Team_Position(:,i) < LB(i));

        violations(team) = violations(team) + abs(sum(LB(i)-
New_Team_Position(indexes,i)));

```

```

New_Team_Position(indexes,i) = LB(i);
indexes = find(New_Team_Position(:,i) > UB(i));

violations(team) = violations(team) + abs(sum(New_Team_Position(indexes,i)-
UB(i)));
New_Team_Position(indexes,i) = UB(i);

end
elseif (opBounds == 2)
for i = 1:Nvars
indexes = find(New_Team_Position(:,i) < LB(i));
violations(team) = violations(team) + abs(sum(LB(i)-
New_Team_Position(indexes,i)));
New_Team_Position(indexes,i) = LB(i) +
(rand(length(indexes),1)*0.1*matInterval(i));
indexes = find(New_Team_Position(:,i) > UB(i));
violations(team) = violations(team) + abs(sum(New_Team_Position(indexes,i)-
UB(i)));
New_Team_Position(indexes,i) = UB(i) -
(rand(length(indexes),1)*0.1*matInterval(i));
end
elseif (opBounds == 3)
for i = 1:Nvars
indexes = find(New_Team_Position(:,i) < LB(i));
violations(team) = violations(team) + abs(sum(LB(i)-
New_Team_Position(indexes,i)));
New_Team_Position(indexes,i) = LB(i) +
abs(rem(New_Team_Position(indexes,i), matInterval(indexes, i)));

indexes = find(New_Team_Position(:,i) > UB(i));
violations(team) = violations(team) + abs(sum(New_Team_Position(indexes,i)-
UB(i)));
New_Team_Position(indexes,i) = UB(i) -
abs(rem(New_Team_Position(indexes,i), matInterval(indexes, i)));

end
end
end

```

```

Teams_Positions{team} = New_Team_Position;

%% Evaluate New_Team_Position Position
for idx = 1:N
    Teams_Positions_Evaluation(idx, team) =
ObjectiveFunction(New_Team_Position(idx,:)); % + violations;
end
NFE = NFE + N;

end % for i = 1:N_Teams

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% COMMAND CENTER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Select for next gen the best solutions by row
for idx = 1:N
    [sorted, indices] = sort(Teams_Positions_Evaluation(idx, :)); % Sort the row position
idx

    CurrentOFV (idx) = sorted(1); % Get the minimum value
    CurrentPosition (idx,:) = Teams_Positions{indices(1)}(idx, :); % Get the solution
from Team (indices(1)) that gave the minimum value

    Teams_Ranking (idx, :) = indices;
end

Shift = LocalBestPosition - CurrentPosition;

%% Acummlate the historical ranks
for (team = 1:N_Teams)
    if (std(Teams_Positions_Evaluation(:, team)) == 0) % ignore clones
        Teams_Ranking(:, team) = Inf;
    end;
end;

```



```
Teams_OFV = Teams_OFV + mean(Teams_Ranking, 1) + violations; % average rank of
each team
```

```
HistoryRanks (Iter-1, :) = mean(Teams_Ranking, 1);
```

```
%% Update Local Best
```

```
improved = CurrentOFV < LocalBestOFV;
```

```
[sorted2, indices2] = sort(CurrentOFV);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% ACCEPT WORSE SOLUTIONS IF SEARCH STAGNATED ? Pacc% OF
CHANCE
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if (Stagnation > MaxStagnation-1),
    fprintf(stderr, ' Stagnated!!! Selecting non-improved solutions.\n');
    indexes = find( (improved + (rand(N, 1) < Pacc)) > 0) ;
    Stagnation = 0;
else
    indexes = find( improved ) ;
end;
```

```
LocalBestOFV(indexes) = CurrentOFV(indexes);
LocalBestPosition(indexes,:) = CurrentPosition(indexes,:);
```

```
%% Update Global Best
```

```
[GlobalBestOFVNew,index] = min(LocalBestOFV);
if GlobalBestOFVNew < GlobalBestOFV
    GlobalBestOFV = GlobalBestOFVNew;
    GlobalBestPosition = repmat(LocalBestPosition(index,:),N,1);
end
```

```
meanOFV = mean(Teams_Positions_Evaluation, 1);
stdOFV = std(Teams_Positions_Evaluation);
```

```

[sortedMeans, indices] = sort(meanOFV);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% TIME TO UPDATE THE FIRMWARE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (rem (Iter, Command_Center_Iter) == 0)

    %% Update FITNESS in GPLab
    for i = 1:N_Teams

        GPCConfig.popu.chrom{i}.fitness = Teams_OFV(i);

        %%% Clear the ranks
        Teams_OFV(i) = 0;

    end

    %%% GENERATE A NEW FIRMWARE TO REPLACE THE WORST ONE
    [GPCConfig.popu, dummy] = gpols_mainloop(GPCConfig.popu, 0, 0, [],
GPCConfig.opt);

    for i = 1:N_Teams
        Firmwares{i} = tree_stringrc( GPCConfig.popu.chrom{i}.tree, 1,
GPCConfig.symbols );
    end

end %% Command_Center_Iters

%% Print Result
Avg = mean(CurrentOFV);
Worst = max(CurrentOFV);
reference = min(CurrentOFV);

Restart = 0;

if (Iter==1)|| (rem(Iter, ReportLag)==0)
    fprintf(stderr, 'Iter: %d | F(x) Best: %.3e | Avg: %.3e | Worst: %.3e | Std-dev: %.3e\n',
Iter, GlobalBestOFV, Avg, Worst, std(CurrentOFV));

```

```

    if (Toolbox)
        str = [cellstr(get(setup.handles.txt_LOG, 'String')); sprintf('Iter: %d | F(x) Best: %.3e
| Avg: %.3e | Worst: %.3e | Std-dev: %.3e', Iter, GlobalBestOFV, Avg, Worst,
std(CurrentOFV))];

        set(setup.handles.txt_LOG, 'String', str);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %% PLOT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        semilogy(StatsCurvesOFV, 'LineWidth', 2)

        % Turn on the grid
        grid on

        legend('Global Best','Current Best','Current Average','Current Median','Current
Worst');

        % Add title and axis labels
        title(sprintf('Value: %.3e   Evaluations: %d', GlobalBestOFV, NFE))
        xlabel('Iterations')
        ylabel('log10 (Objective Function Value)')
        drawnow
    end
end

if ( std(CurrentOFV) < ConvThres ) %% Shall we restart due to convergence ?
    Restart = 1;
elseif ( abs(GlobalBestOFV - PreviousGlobalBestOFV) / abs(PreviousGlobalBestOFV)
< 1e-4 ) %% Needs better improvement
    Stagnation = Stagnation + 1;
else
    Stagnation = 0;
end

if (Restart == 1)

```

```

if Restart == 1
    fprintf(stderr, 'CONVERGENCE DETECTED AT ITER=%d!!!\n', Iter);
else
    fprintf(stderr, 'STAGNATION DETECTED AT ITER=%d!!!\n', Iter);
end

Stagnation=0;

GPConfig = InitGPConfig(N_Teams, maxtreedepth, use_known_perturbation);

for i = 1:N_Teams
    Firmwares{i} = tree_stringrc( GPConfig.popu.chrom{i}.tree, 1,
GPConfig.symbols );
end

fprintf(stderr, ' ## NEW SAMPLE SIZE=%d\n\n', N);

%%%%%% GENERATE INITIAL LARGE (N * N_Teams) SAMPLE FROM THE
LANDSCAPE
%% Initial Position and Velocity
len = N*N_Teams;
CurrentPosition = zeros(len,Nvars); % Initial Position
for i = 1:Nvars
    CurrentPosition(:,i) = randraw('unif',[LB(i),UB(i)],len,1);
end

%% Evaluate Initial Position
CurrentOFV = zeros(len,1); % OFV Value
for i = 1:len
    CurrentOFV(i) = ObjectiveFunction(CurrentPosition(i,:));
end
NFE = NFE + len;

%% Select the N best solutions
[sorted, indices] = sort(CurrentOFV);

%% Update Local Best
LocalBestPosition = CurrentPosition( indices(1:N), : ); % Local Best

```



```

fit=sum((x).^2);
end

```

```

%This function is used to mimic diffusion process, and creates some
%new points based on Gaussian Walks.

```

```

%*****
%The input function is:
%Point: the input point which is going to be diffused
%S: structure of problem information
%g: generation number
%BestPoint: the best point in group
%=====
%The output function is:
%createPoint: the new points created by Diffusion process
%fitness: the value of fitness function
%*****

```

```

function [createPoint, fitness] = Diffusion_Process(Point,S,g,BestPoint)
%calculating the maximum diffusion for each point
NumDiffiusion = S.Maximum_Diffusion;
New_Point = Point;

%Diffusing Part*****
for i = 1 : NumDiffiusion
%consider which walks should be selected.
if rand < S.Walk
GeneratePoint = normrnd(BestPoint, (log(g)/g)*(abs((Point - BestPoint))), [1 size(Point,2)]) + ...
(randn*BestPoint - randn*Point);
else
GeneratePoint = normrnd(Point, (log(g)/g)*(abs((Point - BestPoint))),...
[1 size(Point,2)]);
end
New_Point = [New_Point;GeneratePoint];
end
%check bounds of New Point
New_Point = Bound_Checking(New_Point,S.Lband,S.Uband);
%sorting fitness
fitness = [];
for i = 1 : size(New_Point,1)
fitness = [fitness;feval(S.Function_Name,New_Point(i,:))];
end
[fit_value,fit_index] = sort(fitness);
fitness = fit_value(1,1);
New_Point = New_Point(fit_index,:);
createPoint = New_Point(1,:);
%=====
end

```

```

%*****%
% Author: Hamid Salimi
% Last Edited: 8/16/2014
% Email:salimi.hamid86@gmail.com (or) h.salimi@ut.ac.ir
% Reference: Stochastic Fractal Search: A Powerful Metaheuristic Algorithm%
% Published in Knowledge-Based Systems - Journal - Elsevier.
% Please refer to mentioned journal in term of citation.
% The first online draft of journal paper can be found at below link:
% http://www.sciencedirect.com/science/article/pii/S0950705114002822

```

```

% You have to refer SFS code provided any use of SFS's part in your codes %
% If you encounter any problems or bugs in the SFS code, you are welcome %
% to contact me. %
%*****%
clear all
close all
clc
%Stochastic Fractal Search-----
% The structure S contains all the parameters for the SFS algorithm
%-----
%Initializing Stochastic Fractal Search Parameters*****
% SFS has three main parameters along with an optional parameter:
% 1- Population size considered as Start_Point
% 2- Maximum generation considered as Maximum_Generation
% 3- Maximum Diffusion Number (MDN) considered as Maximum_Diffusion
% Optional parameter: Choosing diffusion walk considered as Walk
S.Start_Point = 50;
S.Maximum_Generation = 400;
S.Maximum_Diffusion = 2;
S.Walk = 1; % *Important
%-----
%*Please Note:
%S.Walk = 1 ----> SFS uses the first Gaussian walk(usually SIMPLE Problems)
%S.Walk = 0 ----> SFS uses the second Gaussian walk(usually HARD Problems)
%You can also write:
%S.Walk = 0.75 ----> SFS uses the first Gaussian walk, with probability
%of 75% which comes from uniform, and SFS uses the second Gaussian walk
%distribution with probability of 25% .
%Generally, to solve your problem, try to use both walks.
%If you want to use the second Gaussian walk please increase the maximum
%generation
%-----
%*****
%Initializing Problem*****
% To initialize your problem, you need to set three parameters:
% 1- Set the name of your function in Function_Name
% 2- Set the dimension of your problem in Ndim
% 3- Set the lower and upper vector bounds in Lband and Uband respectively
% Note: For easy implementation, your functions should take one vector
% & get back one fitness value.
S.Function_Name = 'DeJong';
S.Ndim = 30;
S.Lband = ones(1, S.Ndim)*(-100);
S.Uband = ones(1, S.Ndim)*(100);
%*****
%Plotting Part*****
%If you want to plot the problem in 3D, set it to 1. Your dimension problem
%should be upper than 2, else an error will be occurred.
S.plot = 1;
%Note: plotting the result consumes the time.
%*****
%Printing Result*****
%if you want to see the best result in each iteration, set it to 1.
S.ShowResult = 0;
%Note: printing the result causes consuming time.
%*****
%Start Stochastic Fractal Search*****
%compute the time of finding solution
StartOptimiser = tic;

```

```
[pbest, fbest, F] = Stochastic_Fractal_Search(S);
EndOptimiser = toc(StartOptimiser);
%*****
%Print Final Results*****
fprintf('The time of finding solution is: %f\n', EndOptimiser);
display('The best solution is:');
pbest
display('The value of the best fitness function is:');
fbest
%*****
```