26th ABCM International Congress of Mechanical Engineering
November 22-26, 2021. Florianópolis, SC, Brazil

COB-2021-0029
METAHEURISTIC APPROACHES TO OPTIMIZE A FUZZY-CONTROLLED VEHICLE NAVIGATION

Luiza Scapinello Aquino
Yan Lieven Souza Lúcio
Electrical Engineering Graduate Program (PPGEE), Federal University of Parana (UFPR), Curitiba, PR, Brazil
luiza.scapinello@ufpr.br
yanlieven@ufpr.br

José Henrique Kleinubing Larcher
Mechanical Engineering Graduate Program (PPGEM), Pontifical Catholic University of Paraná (PUCPR), Curitiba, PR, Brazil
jhklarcher@gmail.com

Viviana Cocco Mariani
Mechanical Engineering Graduate Program (PPGEM), Pontifical Catholic University of Paraná (PUCPR), and Department of Electrical Engineering, Federal University of Paraná (UFPR), Curitiba, PR, Brazil
viviana.mariani@pucpr.br

Leandro dos Santos Coelho
Electrical Engineering Graduate Program (PPGEE), Federal University of Parana (UFPR), Curitiba, PR, Brazil, and Industrial and Systems Engineering Graduate Program (PPGEPS), Pontifical Catholic University of Parana (PUCPR)
leandro.coelho@pucpr.br

Abstract. *A metaheuristic optimization approach is used in the tuning of a fuzzy control method for a vehicle navigation simulation. The objective is, given a map containing obstacles, to guide the vehicle to a set target. Since the only data available during the course is the distance from the nearest obstacles and the angle to the destination, it is only possible to control the vehicle's velocity in each possible direction. Unlike more simple control techniques, Fuzzy logic controllers (FLC) allow to make an easy control structure, while at the same time having satisfactory efficiency, given that fuzzy inference systems are capable of representing information from complex or uncertain contexts, like non-linear applications such as the one in this work. However, a deficiency in FLCs is the fact that it is a difficult task to tune the control parameters without previous experience in the manipulation of the model. The optimization of the control parameters can be complex and time-consuming due to the non-linearities present in the model. Evolutionary algorithms and swarm intelligence paradigms, collections of metaheuristic paradigms, can be appropriate to overcome this lack of model knowledge, since they are well known for diversifying the range of solutions, including the optimal one, in the search space, as well as having the ability to use available information from the solution candidates to intensify a search. The contribution of this paper is to compare the performance of three stochastic optimization metaheuristics including Particle Swarm Optimization (PSO), Differential Evolution (DE) and Jaya Optimization to improve the FLC design parameters. The obtained results were promising in terms of the performance indexes in the simulations, they also reveal that all three metaheuristic algorithms are capable of achieving favorable performances for the off-line tuning of the FLC. Further experiments will be made in the future to check practicality with the use of other metaheuristics to FLC tuning.*
Keywords: *Fuzzy Control, Particle Swarm Optimization, Differential Evolution, Jaya Optimization, Vehicle Navigation Simulation*

1. INTRODUCTION

Fuzzy logic controllers (FLCs) have become a popular method of controlling non-linear systems, whose mathematical models are difficult to achieve. This acclaim is related to FLCs' capacity of boosting these kinds of systems' flexibility and robustness, especially when presented with some uncertainty (Sarabakha *et al*., 2019). The problem with FLCs is related to their high computational and memory space requirement, a consequence of their superior performance advantage (Talib *et al*., 2020).

Due to its characteristics, an FLC is appropriate to accomplish the task of navigating a vehicle on a random-made map. In order to function properly, FLCs need well designed heuristic control rules and fuzzy and logic sets. The first one represents rules in linguistic terms, and the others evaluate those rules (Feng, 2006). Poorly designed rules may lead to

the controller becoming unable to correctly regulate the model. A high number of rules normally correlates with better performance, nevertheless, it also is related to a higher computational cost (Talib *et al*., 2020). In this paper, we focus on simplifying the rules, consequently diminishing the computational cost, but still maintaining an acceptable performance, which is measured by the controller's completeness, consistency, and continuity (Masiala, 2010).

Metaheuristic optimization algorithms are problem-solving methods, whose prime objective is to achieve a reasonable solution at a relatively small and acceptable computation time for complex optimization problems, those which usually have a fail rate among classical approaches, or cannot even be applied (Salcedo-Sanz, 2016), normally NP-Hard optimization problems. Metaheuristics use stochastic operations, mimicking a natural, principle, being either physical or biological, in research on a recurring basis, alike an optimization process, thus making it possible to modify some of the initial candidate solutions, which are generated randomly. Some elements may interfere in the results, being the most common ones, the type of problem model, which metaheuristic algorithm is used, as well as the size of the search space, and the computational capacity (Bandaru and Deb, 2016).

The contribution of this paper is to compare the performance of three different metaheuristic optimization approaches, being those the evolutionary approach differential evolution (DE) and two swarm intelligence approaches, particle swarm optimization (PSO) and Jaya optimization, to improve the optimization of a FLC for a vehicle navigation simulation provided a limited search space of control hyperparameters. Although metaheuristics approaches are growly being used in the optimization of a FLC, the ones proposed are less used in the literature, and the aim of this paper is to prove its efficiency.

The structure of the remainder of this paper is organized as follows. The fundamentals of FLC, the vehicle navigation process, as well as a brief understanding of the metaheuristics are presented in Section 2. Sequentially, Section 3 grants the adopted methodology and the optimization procedure. In Section 4, the optimization and control results are described and discussed. Lastly, the conclusion and future scopes are given in Section 5.

2. BRIEF REVIEW OF THEORIC FUNDAMENTALS

Fuzzy systems are a known field in computational intelligence. It focuses on situations in which the value of a given data is not certain being either true or false, but a mid-area between those. FLC is considered a heuristic approach, meaning it does not need an exact mathematical function of the problem, rather it uses the knowledge of the situation in other to find a solution. Therefore, FLCs perform well with imprecise inputs and nonlinearity, also these kinds of controllers are better for environments with great disturbance, consequently they are most used in control purposes that cannot be well handled by classical control methods (Silva and Pinto, 2018).

The process of an FLC is based on its flexible set of if-then rules, which are typically formulated in linguistic terms. The rules are the linguistic relations between the system's input and output, moreover each rule gives as an output a fuzzy set (Eltamaly, 2018). The first phase of an FLC is known as fuzzification, in which the problem's input variables are mapped into suitable linguistic values, in other words, the system's numerical input variables are converted into membership functions. Using more than one rule increases conversion efficiency. The second phase of an FLC is the process in which all the fuzzy sets from the rules are combined into one only fuzzy set. As the output of the controller is now a fuzzy set, the process of defuzzification is required, that is, the conversion of the inferred fuzzy output into a nonfuzzy (also called crisp) control action is needed (Ahmadian, 2001). Figure 1 illustrates the representation of an FLC.

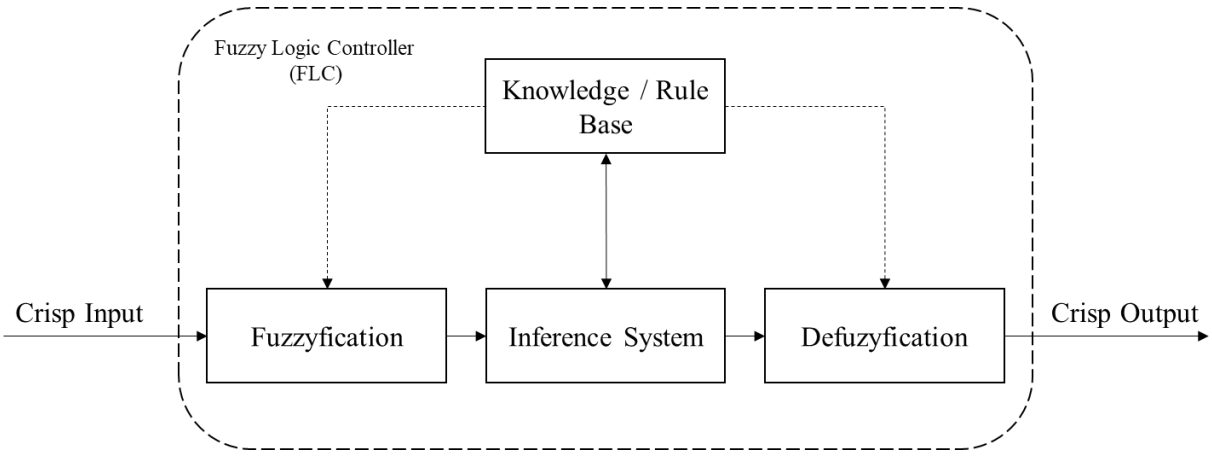

Figure 1. Representation of an FLC

To optimize the FLC, its parametrization can be structured based on metaheuristics approaches, such as DE, PSO and Jaya. These are metaheuristic methods are usually based on natural, physical, or biological principles and attempt to imitate them at a structural level throughout some operators, they can be divided into two different approaches based on

its characteristics. Evolutionary algorithms (EAs) are metaheuristics inspired in aspects of evolution in nature, such as the survival of the fittest, reproduction and genetic mutation, its fundamentals are mostly based in Darwin's evolution theory, while swarm intelligence (SI) algorithms mimic group behavior and/or interactions of living organisms in colonies (Dhiangra *et al*., 2020).

DE algorithms use population-based stochastic search and optimization techniques to solve global optimization problems iteratively. DE is an EA that operates with weighted differences between solution vectors to change the population (Rout *et al*., 2013). In a basic DE, a population of candidate members, each being a sequence of "genes", are created at random, then, for each member, one at a time, a challenger is developed. If the challenger has superior aptitude, it will replace the weakest member in the next generation, hence composing a generation with greater average fitness (Mayer *et al*., 2005). Figure 2 represents a simple pseudocode for a DE.

In a DE algorithm, a population of size $\boldsymbol{N}$ and parameter vectors of dimension $\boldsymbol{D}$ are first provided to initialize the algorithm, in which each parameter vector indicates an individual of the population (Price et al., 2005). A defined maximum number of generations $\boldsymbol{tmax}$ is performed in order to obtain the optimal global solution. After this initialization of the parameter vectors, the DE algorithm cycles to perform the mutation steps with differential vectors, recombination or crossover and selection. In this way, these steps are repeated in subsequent generations and continue until a stopping criterion is satisfied, normally this being the maximum number of generations reached. Figure 2 represents a pseudocode for a generic DE algorithm.

Pseudocode for Differential Evolution (DE)

```
Initialize population size (N), scale factor (F) and crossover ratio (CR)
Randomly generate the members of the population N
while stop criteria not reached do
    Assess fitness values of all members of the population
    for i = 1, 2, ...,N do
        Generate integer random number rand_i between 1 and search space dimension D
        for j = 1, 2, ..., D do
            if rand(0, 1) ≤ CR or j = rand_i then
                u_ji = v_ji = x_jr1 + F.(x_jr2 − x_jr1)
            else
                u_ji = x_ji
            end if
        end for
        Assess experimental vector u_i
        if f(u_i) ≤ f(x_i) then
            Change x_i for u_i
        end if
    end for
end while
```

Figure 2. Pseudocode of a DE algorithm

As seen in Fig 2, if the integer and uniformly distributed random number $\boldsymbol{rand_i}$ is less or equal than the crossover factor, the challenger $\boldsymbol{u_{ji}}$ is created based on other two population members $\boldsymbol{x_{jr1}}$ and $\boldsymbol{x_{jr2}}$, defined by the randomly created numbers $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$, otherwise the challenger is the original gene $\boldsymbol{x_{j,i}}$. Then both the original member and the challenger are evaluated by the fitness function, and the one with the better score is kept and the other replaced.

PSO (Kennedy and Eberhart, 1995) is a SI algorithm in which a population (swarm) of particles (candidate solutions) move in the search space looking for the optimum point of a function, it was inspired by the coordinated motion of animals living in society. In a PSO each particle is associated with two complementary vectors, representing the position and velocity of the particle through the search space. The particles keep track of the best position previously obtained. The positions of the particles are classified between the personal best $\boldsymbol{pbest}$ or local best $\boldsymbol{lbest}$, when talking about only one particle. The best position of a particle, when compared to all the particles in the cluster, is the global best $\boldsymbol{gbest}$ (Cheng-Yuan *et al.*, 2010).

A particle of a swarm of size $\boldsymbol{N}$ moves in a search space of dimension $\boldsymbol{D}$. The position $\boldsymbol{x}$ and speed $\boldsymbol{v}$ of each particle are represented by vectors of the same size as the dimension of the search space, the same occurs with the vectors of best personal position and best global position $\boldsymbol{x_{pbest}}$ and $\boldsymbol{x_{gbest}}$, respectively. Random constants $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are uniformly distributed within the range of zero to one and are created to calculate the velocity and position of the particles. The positive constants $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$ are called acceleration coefficients, the first refers to the cognitive parameters of the particle, while the second deals with the social ones, their job is to accelerate the particles to the best personal and global positions of the cluster. The adjustment of the coefficients $\boldsymbol{c_{1,2}}$ are important for determining the exploitation and exploitation of

the algorithm. When small values for $c_{1,2}$ are used, the particles move away from the target region, while high values provide an abrupt movement towards or beyond the target region (Eberhart and Shi, 2001). In Fig. 3 it is possible to understand how this algorithm works through a pseudocode.

Algorithm 2: Particle Swarm Optimization (PSO)

1: Initialize the N particles, maximum velocity limit v_{max} and acceleration coefficients c_1 e c_2
2: Randomly initialize the velocity and position of the N particles
3: The best local position of a particle is attributed to the initial position of the particle
4: Assess the fitness values (*fitness f ()*) of all members of the swarm
5: Calculate the best particle in the swarm as x_{gbest}
6: **while** stop criteria not reached **do**
7: **for** i = 1, 2, ...,N **do**
8 Calculate the velocity of the i[th] particle: $\boldsymbol{v_i}(t+1) = \boldsymbol{v_i}(t) + c_1\boldsymbol{r_1}(x_{pbest_i}(t) - x_i(t)) + c_2\boldsymbol{r_2}(\boldsymbol{x_{gbest_i}}(t) - \boldsymbol{x_i}(t))$
9: Calculate the position of the i[th] particle: $\boldsymbol{x_i}(t+1) = \boldsymbol{x_i}(t) + \boldsymbol{v_i}(t+1)$
10: Assess the fitness values of each member of the swarm
11: **if** $f(x_i) \leq f(x_{pbest\_i})$ **then**
12: $x_{pbest\_i} = x_i$
13: **end if**
14: **if** $f(x_i) \leq f(x_{gbest\_i})$ **then**
15: $x_{gbest\_i} = x_i$
16: **end if**
17: Update the position and speed of the i[th] particle to x_i and v_i
18: **end for**
19: **end while**

Figure 3. Pseudocode of a PSO algorithm

The Jaya optimization algorithm (Rao, 2016) is a faster heuristic search algorithm based on simple swarm intelligence. It is established on the idea of defeating the weakest opponent and finding the targeted path to victory, therefore, this optimization algorithm requires that the common control parameters move towards the best solution avoiding the worst (Mishra and Ray, 2016). In a Jaya algorithm, the superiority between the solutions is decided by a non-dominance rank and the value of the estimated density parameter. The solution with the highest rank and with the highest density value is chosen as the best solution. A similar process occurs to find the worst solution (Rao et al., 2017).

After initialization, the best and worst candidate in the current generation are found. Next, the solutions are modified according to Jaya's main equation. In this equation $\boldsymbol{X_{j,k,i}}$ represents the value for the $\boldsymbol{j^{th}}$ variable of the $\boldsymbol{k^{th}}$ candidate during the $\boldsymbol{i^{th}}$ iteration, moreover, $\boldsymbol{r_{1,j,i}}$ and $\boldsymbol{r_{2,j,i}}$ are randomly generated numbers uniformly distributed between one and zero. The score for $\boldsymbol{X'_{j,k,i}}$ is acceptable by proving its value on the fitness function and compared to $\boldsymbol{X_{j,k,i}}$. The most appropriate values of the function are stored at the end of each iteration and become input for the next one, in order to obtain the final optimal solution (Pradhan, 2019). Figure 4 represents a pseudocode for a Jaya optimization algorithm.

Algorithm 3: Jaya Optimization

1: Initialize population size and designed variables
2: **while** stop criteria not reached **do**
3: Search individuals for best and worst solution
4: Modify the solution with $\boldsymbol{X'_{j,k,i}} = \boldsymbol{X_{j,k,i}} + \boldsymbol{r_{1,j,i}}[\boldsymbol{X_{j,best,i}} - |\boldsymbol{X_{j,k,i}}|] - \boldsymbol{r_{2,j,i}}[\boldsymbol{X_{j,worst,i}} - |\boldsymbol{X_{j,k,i}}|]$
5: Update previous solution if $\boldsymbol{X'_{j,k,i}} > \boldsymbol{X_{j,k,i}}$
6: **end while**

Figure 4. Pseudocode of a Jaya optimization algorithm

3. DESCRIPTION OF THE CASE STUDY: VEHICLE NAVIGATION

To test the efficiency of the PSO, Jaya and DE optimizations, the chosen problem was a vehicle navigation simulation, in which the objective is to guide an agent to a set location in a randomized map. The movement of the agent is controlled

in each simulation iteration by setting its velocity in each possible direction, calculated by using its distance to the nearest obstacles and the angle to the destination. Figure 5 shows the agent's route to the objective in a random map.

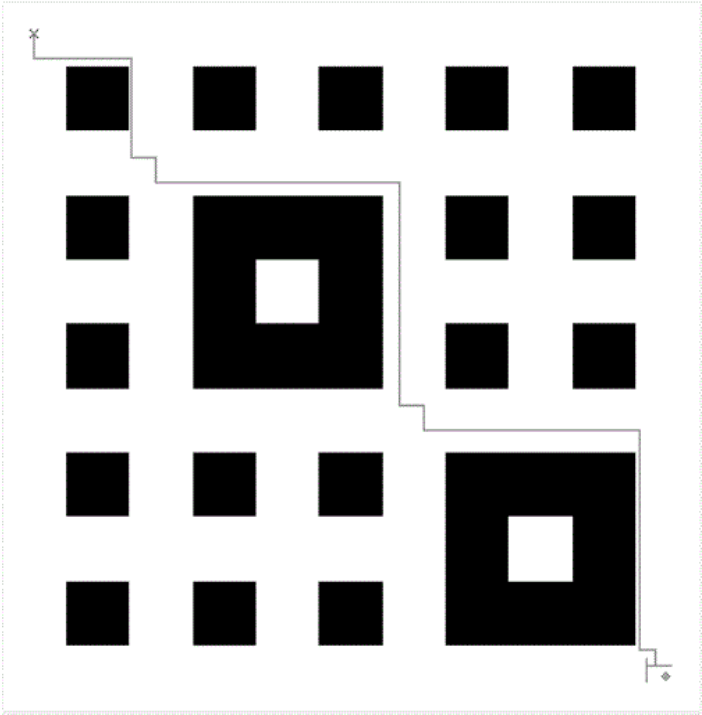

Figure 5. Agent (top left) route to the objective (bottom right) in a randomized map.

The controller used for this application is an FLC, due to how well fuzzy logic can deal with nonlinearities such as the ones present in this model. The FLC receives the angles to the objective (first input) and the distances to the nearest obstacles (second input) in each direction (north, east, south, and west) to compute the output velocity. Each input has two membership functions, each representing "good" or "bad" values, and the output has three, for "low", "moderate" or "high" values. The output functions are shown in Figure 6. The vehicle after calculate the velocity in each of the four direction moves into the one with the higher speed value.

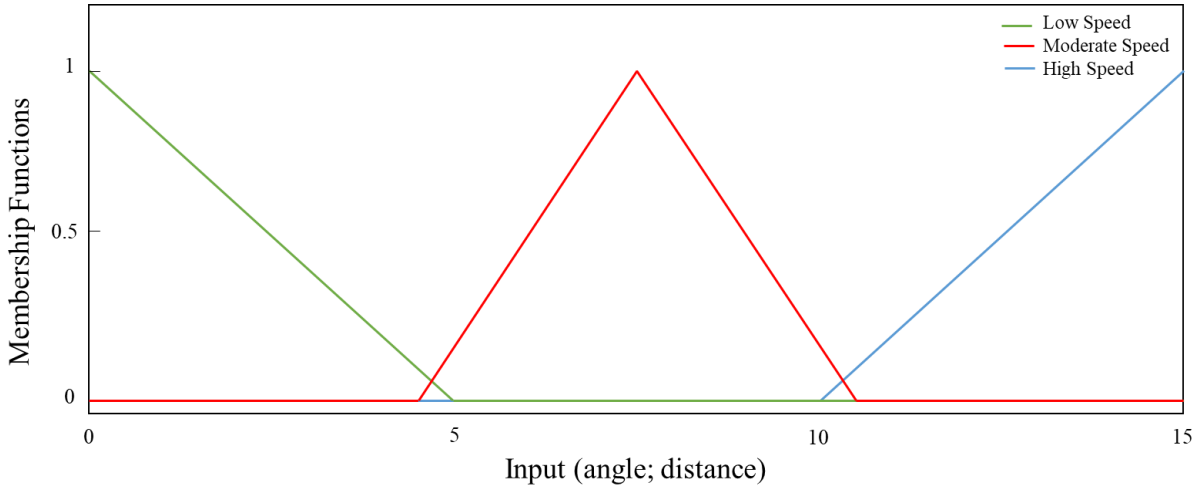

Figure 6. Output triangular membership functions.

The FLC is also composed of three rules in the IF <antecedent> THEN <consequent> format:

1. *If angle is "good" and "distance" is "good" then velocity is "high"*
2. *If angle is "bad" then velocity is "low"*
3. *If angle is "good" and "distance" is "bad" then velocity is "moderate"*

For optimizing the FLC, it was chosen to modify 4 values in the output pertinence functions. The solution vector is shown in Figure 7, where $\mathbf{P_1}$ is the right end of "lowSpeed", $\mathbf{P_2}$ is the left end of "highSpeed" and $\mathbf{P_3}$ and $\mathbf{P_4}$ the left and right end values of "midSpeed".

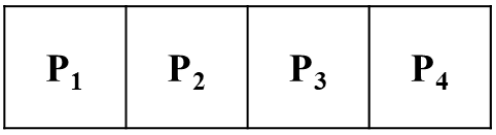

Figure 7. Solution Vector.

For comparing controllers, it was defined that for the same map, an FLC is better than another if his final distance to the objective at the end of 60 movement iterations is lower than that of the other controller. As each map is created randomly, the fitness function value for a single controller is defined as the sum of the final distances to the objective in 100 maps, as shown in Eq. 1, in which the $\boldsymbol{FD_i}$ is the final distance of the FLC for map number $\boldsymbol{i}$.

$$\boldsymbol{fitness}(x) = \sum_{\boldsymbol{i}=1}^{100} \boldsymbol{FD_i}(x), \quad (1)$$

Each algorithm was run 50 times, with 50 as the population size and 125 as the number of generations. The results and comparison for each algorithm are presented in the next section.

4. RESULTS AND DISCUSSION

Table 1 shows the statistical results of the fitness function values for each one of the three metaheuristics studied. In this table, the minimum, maximum, mean, median and standard deviation values for the 50 runs with 100 different maps of the optimizers are presented in order to compare the different techniques.

The results are indeed similar with each other, although a better performance is seen in the Jaya optimization algorithms in all criteria except for the mean statistic, in which PSO has a better performance. The optimal values are represented by the ones with smaller error. Overall, these swarm intelligence algorithms showed to have accomplished the task in better terms than the evolutionary algorithm. The best result was achieved by the Jaya optimization algorithm, since its minimum and mean error were the smallest ones among the studied algorithms.

Table 1. Experimental results for the FLC of a vehicle navigation.

			Minimum	Maximum	Mean	Median	Standard Deviation
Jaya	$\mathbf{P_1}$	Low Speed	2.5000	7.5000	4.5076	4.0201	1.8314
	$\mathbf{P_2}$	High Speed	7.5000	12.5000	9.4995	9.3640	1.8124
	$\mathbf{P_3}$	Left Mid Speed	2.5000	6.2500	4.3674	4.3782	1.4486
	$\mathbf{P_4}$	Right Mid Speed	8.7500	12.5000	10.5308	10.6268	1.1945
	Error	Error	2044.9297	4559.6287	3404.6041	3416.3858	611.6227
PSO	$\mathbf{P_1}$	Low Speed	2.6062	6.9086	4.1455	3.9093	1.2515
	$\mathbf{P_2}$	High Speed	7.5486	10.4105	8.9476	8.7072	0.8208
	$\mathbf{P_3}$	Left Mid Speed	2.5000	5.1364	3.6349	3.6396	0.8882
	$\mathbf{P_4}$	Right Mid Speed	8.7500	10.5879	9.8379	10.0160	0.6547
	Error	Error	2396.9593	6374.5550	3698.9739	3766.0049	666.4131
DE	$\mathbf{P_1}$	Low Speed	2.6608	7.5000	4.5501	4.1240	1.6978
	$\mathbf{P_2}$	High Speed	7.5940	12.5000	9.5000	9.3198	1.8005
	$\mathbf{P_3}$	Left Mid Speed	2.5000	6.5051	4.3397	4.6921	1.5264
	$\mathbf{P_4}$	Right Mid Speed	8.7500	12.5000	10.5489	10.5890	1.1958
	Error	Error	2845.4845	6059.8756	3944.7981	3934.2846	700.9432

Figure 8 represents the best FLC for each one of the three algorithms for the vehicle navigation in the map shown on Fig. 5. And it is possible to see the Jaya one having a smoother action than the others. The control signal represents affects the movement speed, consequently its velocity which is a vector calculated with the help of the angle input.

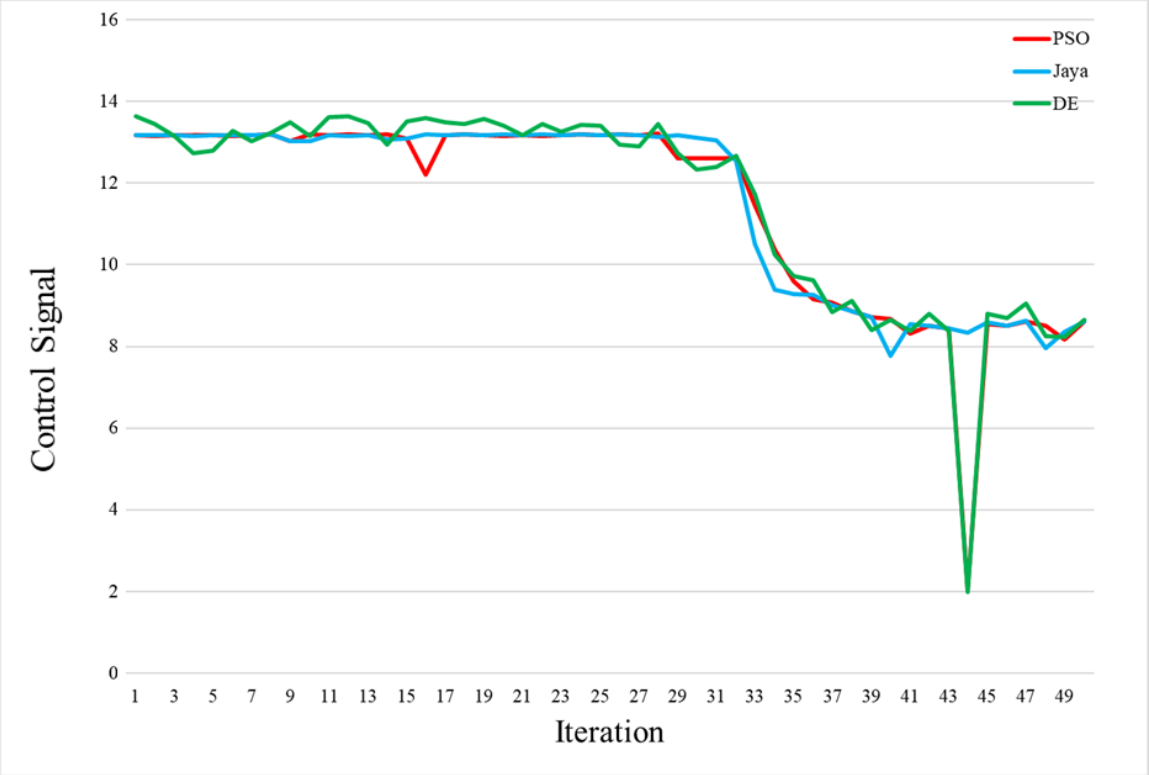

Figure 8. Best FLC of each optimization.

5. FINAL CONSIDERATIONS

As previously mentioned, all metaheuristic algorithms are affected by the problem to be solved and all the data presented by it. All optimization algorithms could complete the task, moreover the most recently created of the three, the Jaya optimization was the most efficient, and DE the only one based on evolutionary approaches was the one with the worst outcome. Further research includes improving the efficiency of the optimizers, as well as trying different metaheuristics for the same problem of FLC tuning.

6. ACKNOWLEDGEMENTS

The authors would like to thank the National Council of Scientific and Technologic Development of Brazil - CNPq (Grants number: 307958/2019-1-PQ and 404659/2016-0-Univ), PRONEX `Fundação Araucária' 042/2018 and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance Code 001.

7. REFERENCES

Ahmadian, M., 2001. “Active control of vehicle vibration”. *Encyclopedia of Vibration*, pp. 37-45.

Bandaru, S., Deb, K., 2016. “Metaheuristic techniques”. *Decision sciences: Theory and practice*, Vol. 220, No. 4598, pp. 693-750.

Cheng-Yuan, L., Yan-Rui, D., Wen-Bo, X., 2010. “Multiple-layer quantum-behaved particle swarm optimization and toy model for protein structure prediction”. In: *Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, Hong Kong, pp. 92-96.

Dhiangra, S., Sowdhamini, R., Cadet, F., 2020. “A glance into the evolution of template-free protein structure prediction methodologies”. *Biochimie*, Vol. 175, pp. 85-92.

Eberhart, R.C., Shi, Y., 2001. “Particle swarm optimization: developments, applications and resources”. In: *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 1, p. 81-86.

Eltamaly, A. M., 2018. "Performance of MPPT techniques of photovoltaic systems under normal and partial shading conditions". *Advances in Renewable Energies and Power Technologies*, Vol. 1, pp. 115-161.

Feng, G., 2006. "A survey on analysis and design of model-based fuzzy control systems". *IEEE Transactions on Fuzzy systems*, Vol. 14, No. 5, pp. 676-697.

Github, 2021. *Neuro-fuzzy-vehicle-controller*, https://github.com/nickgkan/neuro-fuzzy-vehicle-controller/blob/master/control_vehicle.m. Access on 10 March 2021.

Kennedy, J. and Eberhart, R.C., 1995. "Particle swarm optimization". In: IEEE International Conference on Neural Networks, Perth, WA, Australia, Vol. 4, pp. 1942-1948.

Masiala M., 2010. "Self-tuned indirect field oriented controlled IM drive". Ph.D. thesis, Edmonton, Alberta, Canada: Alberta University.

Mayer, D. G., Kinghorn, B. P., Archer, A. A., 2005. "Differential evolution – an easy and efficient evolutionary algorithm for model optimization". *Agricultural Systems*, Vol. 83, pp. 315-328.

Mishra, S., Ray, P. K., 2016. "Power quality improvement using photovoltaic fed DSTATCOM based on JAYA optimization". *IEEE Transactions on Sustainable Energy*, Vol. 7, No. 4, pp. 1672-1680.

Pradhan, C., Bhende, C. N., 2019. "Online load frequency control in wind integrated power systems using modified Jaya optimization". *Engineering Applications of Artificial Intelligence*, Vol. 77, pp. 212-228.

Price, K., Storn, R.M., Lampinen, J.A, 2005. "Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series". First Edition Springer-Verlag: New York, United States of America.

Rao, R. V., 2016. "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems". *International Journal of Industrial Engineering Computations*, Vol. 7, pp. 19-34.

Rao, R. V., Rai, D. P., Balic, J., 2017. "Optimization of abrasive waterjet machining process using multi-objective Jaya algorithm". *Materials Today*, Vol. 5, pp. 4930 – 4938.

Rout, U. K., Sahu, R. K., Panda, S., 2013. "Design and analysis of differential evolution algorithm based automatic generation control for interconnected power system". *Ain Shams Engineering Journal*, Vol. 4, No. 3, pp. 409-421.

Sarabakha, A., Fu, C., Kayacan, E., 2019. "Intuit before tuning: Type-1 and type-2 fuzzy logic controllers". *Applied Soft Computing*, Vol. 81, Article number 105495.

Salcedo-Sanz, S., 2016. "Modern meta-heuristics based on nonlinear physics processes: a review of models and design procedures". *Physics Reports*, Vol. 655, pp. 1-70.

Silva, J. F., Pinto, S. F., 2018. "Linear and nonlinear control of switching power converters". *Power Electronics Handbook*, pp. 1141-1220.

Talib, M. H. N., Ibrahim, Z., Rahim, N. A., Zulhani, R., Nordin, N., Farah, N., Razali, A. M., 2020. "An improved simplified rules fuzzy logic speed controller method applied for induction motor drive". *ISA Transactions*, Vol. 105, pp. 230-239.

8. RESPONSIBILITY NOTICE

The authors are the only ones responsible for the printed material included in this paper.