Hindawi
Journal of Advanced Transportation
Volume 2021, Article ID 5538296, 12 pages
https://doi.org/10.1155/2021/5538296

Research Article

Speed Proportional Integrative Derivative Controller: Optimization Functions in Metaheuristic Algorithms

Luis Fernando de Mingo López,[1] **Francisco Serradilla García**,[1,2] **José Eugenio Naranjo Hernández**,[1,2] **and Nuria Gómez Blas**[1]

[1]*ETSI de Sistemas Informáticos, Universidad Politécnica de Madrid, Madrid, Spain*
[2]*Instituto Universitario de Investigación del Automóvil (INSIA), Universidad Politécnica de Madrid, Madrid, Spain*

Correspondence should be addressed to José Eugenio Naranjo Hernández; joseeugenio.naranjo@upm.es

Received 20 January 2021; Revised 7 July 2021; Accepted 2 October 2021; Published 3 November 2021

Academic Editor: Chi-Hua Chen

Recent advancements in computer science include some optimization models that have been developed and used in real applications. Some metaheuristic search/optimization algorithms have been tested to obtain optimal solutions to speed controller applications in self-driving cars. Some metaheuristic algorithms are based on social behaviour, resulting in several search models, functions, and parameters, and thus algorithm-specific strengths and weaknesses. The present paper proposes a fitness function on the basis of the mathematical description of proportional integrative derivate controllers showing that mean square error is not always the best measure when looking for a solution to the problem. The fitness developed in this paper contains features and equations from the mathematical background of proportional integrative derivative controllers to calculate the best performance of the system. Such results are applied to quantitatively evaluate the performance of twenty-one optimization algorithms. Furthermore, improved versions of the fitness function are considered, in order to investigate which aspects are enhanced by applying the optimization algorithms. Results show that the right fitness function is a key point to get a good performance, regardless of the chosen algorithm. The aim of this paper is to present a novel objective function to carry out optimizations of the gains of a PID controller, using several computational intelligence techniques to perform the optimizations. The result of these optimizations will demonstrate the improved efficiency of the selected control schema.

1. Introduction

Many optimization problems are nondeterministic polynomial-time (NP) or NP-hard, and a high computing power is required when trying to solve them [1, 2]. An NP-hard problem "is a problem where a solution for it is at least as hard as finding a solution for the hardest problem whose solution can quickly be checked as being true. Some NP-hard problems are ones in which a working solution can be checked quickly (NP problems) and some are not. NP-hard problems are also NP problems fit into a label called NP-complete" [3]. Many of the problems arising in present-day applications from scientific fields belong in NP-hard problems: they involve search spaces with many dimensions, they are multimodal or multiobjective, and the optimization functions are hard to compute or are applied on large volumes of data. Classical optimization methods from operation research make it possible to find optimal solutions for complex problems, but are not useful in practice due to their excessive computational load when applied to real-world systems [4]. For NP-hard problems, the time required to solve a problem grows exponentially with respect to the size of the problem, making the exact methods unpractical.

In order to solve the disadvantages of classical trial-and-error methods and mathematical solution search techniques, researchers have proposed various algorithms that mimic natural and artificial phenomena, and black-box optimization benchmark problems are implemented to evaluate the performance of these algorithms.

The research area of metaheuristic search/optimization algorithms has an active development of finding inspiration in nature, especially in social behaviour. Among the most classical models are genetic algorithms [5–7], but there are others based on the social behaviour of ants and ant colony optimization [8–10]; in recent years, a lot of methods based on natural heuristics have been proposed: birds [11, 12], reincarnation [13], zombies [14], bees [15–17], etc. This paper shows the application of 21 algorithms' implementation as follows: particle swarm optimization (PSO) [18], clonal selection algorithm (CLONALG) [19], whale optimization algorithm (WOA) [20], shuffled frog leaping (SFL) [21], differential evolution (DE) [22, 23], cat swarm optimization (CSO) [24], ant lion optimizer (ALO) [25], artificial bee colony algorithm (ABC) [26], firefly algorithm (FFA) [27], grey wolf optimizer (GWO) [28], dragonfly algorithm (DA) [29], grasshopper optimization algorithm (Goa) [30], genetic algorithm (GA) [31], harmony search algorithm (HS) [32], sine cosine algorithm (SCA) [33], moth flame optimizer (MFO) [34], krill herd algorithm (KH) [35], bat algorithm (BA) [36], gravitational-based search (GBS) [37], cuckoo search (CS) [38], and black hole optimization (BHO) [39].

Some of the most representative computational intelligence algorithms today are earthworm optimization algorithm (EWA) [40], monarch butterfly optimization (MBO) [41, 42], moth search (MS) algorithm [43], slime mould algorithm (SMA) [44], elephant herding optimization (EHO) [45], and Harris hawks optimization (HHO) [46], among others [47, 48].

Each of these algorithms has its own peculiarities, and they can be used to solve various categories of optimization problems. The performance of evolutionary algorithms depends on the choice of their control parameters, which must be tuned for each specific problem. Population-based metaheuristic algorithms generally include two processes: exploration and exploitation. The exploration process attempts to analyse a wide variety of solutions and must be random enough to ensure that it covers a large area of the problem space. The exploitation process tries to improve, through a local search, the solutions found in the exploration process. In this phase, the optimizer focuses on the neighborhood of the highest quality solutions found by the exploration process, rather than the entire solution space. In any optimization process, it is essential to find an adequate balance between exploration and exploitation, a balance that depends on the nature of the specific problem to be solved.

Hybrid optimization techniques are available in the state of the art for solving complex engineering problems, which combine several simple algorithms to obtain better optimization efficiency.

In the automotive industry, especially for vehicle design and optimized components, the Butterfly Optimization Algorithm (BOA) is a widely used hybrid optimization technique. Also, other hybridization proposals are available for similar purposes, such as Harris Hawk optimization with simulated annealing (HHOSA), which provides an accelerated convergence. Additionally, a hybridization of the interior search algorithm with the hill climbing algorithm (H-ISA) has been used to optimize structural and mechanical design problems, among others. This algorithm has been proven to work better in those cases than the ant lion optimizer (ALO), gravitational search algorithm (GSA), firefly algorithm (FFA), league championship algorithm (LCA), bat algorithm (BAT), symbiotic imperialist competitive algorithm (ICA), organisms search (SOS), and charged system search algorithm (CSS).

The key element for any of these algorithms is the fitness function, which must meet the following conditions: (a) the function must be clearly defined; (b) the implementation of fitness must be as efficient as possible; since it will be evaluated many times in the optimization process, its efficiency has a strong impact on the overall performance of the algorithm; (c) fitness must provide a quantitative measure of the quality of a given solution; (d) the best individuals must produce the best values of fitness function and vice versa.

The fitness function is dependent on the problem domain. The implementation of this fitness function is the main part when you want to apply a nature-based algorithm to a problem, and it must be built for each specific problem. For certain types of problems, for example, for classification tasks with supervised learning, error metrics, such as the Euclidean distance or the Manhattan distance, are often used. For other optimization problems, the summation of a set of quality indicators related to the application domain can be used.

This paper shows the importance of choosing a right fitness function; it is a key point to get a good performance regardless of the chosen algorithm. Discussion about whether a given algorithm obtains a better result than others is not addressed in this study.

The main highlights of the paper are

(i) The evaluation of different metaheuristics techniques applied to PID controllers

(ii) Showing the importance of choosing the right fitness function (mean squared error is not always the best choice)

(iii) Final fitness function that improves the mean squared error achieving a quasi-perfect fit of desired signal

2. Proportional Integrative Derivative Controller

Proportional integrative derivative controllers are the controllers par excellence, as well as the most studied. This type of controllers is based on the old proportional controls, such as the centrifugal regulator of Watt's machine (1788). One important thing is that knowledge about the process to control is not really needed when using it with a PID controller, which is the reason why such a mathematical model is so broadly used. The PID equation describes how to control the system: it receives an error calculated from the desired/obtained output, using this error to feed the control loop in the next iteration. The aim of the controller is to minimize this calculated error by adjusting the gains of the PID.

The PID controller is based on the relation of three components: the proportional, the integral, and the derivative. The tune of the gains of these components not only generates the behaviour of the output of the controller but also influences the others. Although those parameters are tuned, it is quite difficult to manually define the values of the PID gains that generate the values tending to zero of the performance metrics (establishment times, over-oscillation, settling, etc.). However, the best tuning of the controller tries to get a trade-off among those error metrics, adjusting them according to the requirements of the application.

A PID controller outputs a signal using a weighted sum of three terms:

(i) Proportional: error between the defined setpoint and the actual process value

(ii) Integral: how much error has been accumulated over time

(iii) Derivative: how rapidly the error is changing

This naturally leads to P (Proportional), I (Integral), and D (Derivative) in PID and the three weighting parameters (K_p, K_i, K_d) that need to be computed. By modifying these control parameters, the behaviour of these on the system will be modified, and it will be possible to obtain a controller that satisfies the design criteria established for a specific application.

Each control action is responsible for producing a specific effect on the transitional or permanent regime, or on both at the same time. Therefore, when it comes to achieving better control over the system, it is necessary to know which aspect of the control is to be improved, to modify a specific action and not to disrupt the rest of the system.

One must be aware that proportional action affects both the transitional and the permanent regime, tending to reduce the error in the permanent regime when its action increases. On the contrary, when this action increases the system will tend to increase the oscillations in the process variable.

The integral action affects the permanent regime, annulling the error. The error could be understood as the difference between the reference and the value read by the sensor. If the error is greater than zero, this means that the value read by the sensor will be below the reference, meaning that the control action will increase, while if the error is less than zero, the value read by the sensor will be greater than the reference, and therefore, the action on the system will decrease.

The last control action is the derivative action. This action is mainly focused on the transitional regime, improving the stability of the system. It can also be found in some articles as a predictive action, and this is because this control action is of an anticipatory nature, i.e., it anticipates the behaviour of the system to improve its performance. The main disadvantage of this action is that it amplifies the noise of the signal and saturates the actuators in the event of sudden changes in the set point.

When designing a PID controller, several methods can be found in the literature. One of the most famous methods is the Ziegler–Nichols method [49], which is proposed by Ogata [50]. Other methods can also be found, such as those Chien–Hrones–Reswick methods proposed by [51] and the Cohen-Coon method proposed by [52]. In order to tune the PID controller, using these methods, it will be necessary to know the behavior of the system. To do this, the corresponding actuator will be put into operation and wait until the variable in question is stabilized in a permanent state. Observing the variable to control shape (see Figure 1), the values of the gain K, the delay L, and the time constant T (with $a = L/T$) could be calculated depending on the chosen method (see Tables 1–4).

Some authors focus on improving the response of the PID controllers using metaheuristics, especially in problems with a highly nonlinear behaviour, because, for them, the traditional methods of adjusting the gains are inefficient, obtaining results comparable or superior to conventional techniques [53]. Valluru and Singh [54] show that the efficiency of tuning nonlinear drivers using particle swarm optimization and other bioinspired techniques is superior to the results with traditional tuning techniques. The experimental results show that the overshoot and settling time of a nonlinear PID controller can be improved while maintaining a satisfactory system response. Recent research [55] shows that the performance of a PID control system with gains calculated by a symbiotic organism search algorithm manages to minimize the steady-state error. When the system is subjected to disturbances, this controller is capable of reaching the set point in any situation.

This paper does not take disturbance into account; it only shows the importance of choosing a right fitness function. Talking about exogenous disturbance (input disturbance) that affects model (physical or mathematical) behaviour is most probably assimilated to account for model-uncertainty (inaccuracy), time-varying parameters, perturbation (as wind-gust in aircrafts), actuator unmodelled dynamics, and the same ilk. Input disturbance could be modelled as a constant signal $D(s) = 1/s$, in Laplace domain, or $d(t) = 1$ in time domain, or a sinusoidal signal, or a stochastic process, or something like that.

3. Problem Description

There is an electric vehicle with cameras and sensors for speed, acceleration, LiDaR, etc. (see Figure 2). This vehicle can be driven remotely through the ODBII port (see Figure 3) by adjusting the cruising speed, acceleration, and turning the steering wheel. The problem is to set a cruising speed (higher or lower than the current one) and see the vehicle's response under real conditions.

The cruise control is based on a PID. A manual adjustment of the parameters (K_p, K_i, K_d) could be done, but the idea is to try to find a better solution than the manual operator or expert could achieve. The problem can be formulated as a model of optimization in a three-dimensional space. The optimization process requires an objective function (fitness) to be minimized depending on the parameters (K_p, K_i, K_d). Mean square error is the classical

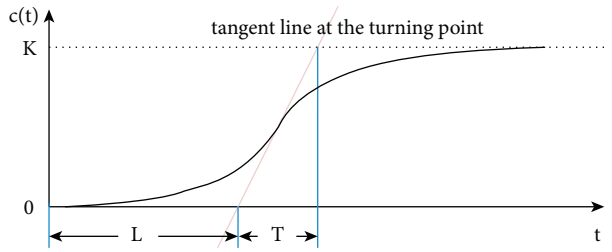

Figure 1: How to compute values corresponding to the gain K, the delay L, and the time constant T over the signal to control. Different methods can be applied to obtain (K_p, K_i, K_d) parameters of the controller (see Tables 1–4).

Table 1: The Ziegler–Nichols tuning rules create a quarter wave decay. This is an acceptable result for some purposes, but not optimal for all applications [49, 50].

Controller	k_p	k_i	k_d
P	$1/a$	0	0
PI	$0.9/a$	$3L$	0
PID	$1.2/a$	$2L$	$L/2$

Table 2: The Chien–Hrones–Reswick autotuning method focuses on set point response and disturbance response (20% overshoot) [51].

Controller	k_p	k_i	k_d
P	$0.3/a$	0	0
PI	$0.6/a$	$4L$	0
PID	$0.95/a$	$2.4L$	$0.42L$

Table 3: The Chien–Hrones–Reswick tuning rules (20% overshoot) [51].

Controller	k_p	k_i	k_d
P	$0.7/a$	0	0
PI	$0.7/a$	$2.3L$	0
PID	$1.2/a$	$2L$	$0.42L$

Table 4: The Cohen-Coon tuning rules (0% overshoot). Note that these rules produce a quarter-amplitude damping response [52].

Controller	k_p	k_i	k_d
P	$1/a(1 + (0.35\tau/1 - \tau))$	0	0
PI	$0.9/a(1 + (0.92\tau/1 - \tau))$	$L(3.3 - 3\tau/1 + 1.2\tau)$	0
PD	$1.24/a(1 + (0.13\tau/1 - \tau))$	0	$L(0.27 - 0.36\tau/1 - 0.87\tau)$
PID	$1.35/a(1 + (0.18\tau/1 - \tau))$	$L(2.5 - 2\tau/1 + 0.39\tau)$	$L(0.37 - 0.37\tau/1 - 0.18\tau)$

Figure 2: Autonomous electric vehicle used in real testing and light detection and ranging sensor (LiDAR) placed on the vehicle windscreen.

fitness function in optimization problems, but there are other approaches depending on the problem to solve. A good performance can be obtained, when dealing with PID, using a linear combination of overshoot, decay, setting time, and steady-state error as proposed by [56, 57]. This paper uses the same mathematical model of the system/process for which the controller described in [56, 57] is employed.

A fractional-order PID controller has better control performance than classical PID controllers. A fractional-order controller includes the capacity of adjusting responses

Figure 3: ODBII interface for controlling the vehicle and visualization of front and rear cameras for remotely controlling speed and direction.

of the control system in time and frequency. This feature grants a better and more robust performance than classical PID controllers. It is not always the case in which a fractional-order PID controller used for integer-order plants will be better than an integer-order PID. However, it has been shown in [58, 59] that the use of a fractional-order PID could make the entire control system perform better. This is because using a fractional one could satisfy 5 robustness criteria $(K_p, K_i, K_d, \mu, \delta)$ at most as compared to the usual classical PID controller which only has 3 parameters to be tuned for 3 robustness criteria. This paper uses a PID controller with precalculated μ and δ values from [56, 57].

The first fitness function f_1, see equation (1), is based on the mean square error (MSE), that is, the average squared difference between the estimated/target speed signal and the actual speed signal:

$$f_1 = \sqrt{\sum_t (\text{target}(t) - \text{signal}(t))^2}. \tag{1}$$

The second fitness function f_2, see equation (2), uses a lineal combination of the overshoot and decay of the signal to avoid large peaks and oscillation in the control response, with precomputed values of α and β [56, 57]:

$$f_2 = \alpha\Omega + \beta\Delta. \tag{2}$$

And the third one f_3, see equation (3), uses a lineal combination of the overshoot, decay, setting time, and steady-state error of the signal to avoid large peaks and oscillation in the control response and get an almost fit response to the original signal:

$$f_2 = \alpha\Omega + \beta\Delta + \gamma\Gamma + \theta\Theta. \tag{3}$$

The overshoot Ω can be defined as the gap between the outputs of the controller from the set point steady-state value. Following [50], a controller gain tuning is considered good in the case of an overdamped response with a minimum overshoot, according to the application. The value of the overshoot, see (4), is the difference between the system output and the target value (with $\alpha = 10$ and $\beta = 0.9$, to obtain better results):

$$\Omega = \begin{cases} \max(\text{signal}(t)) - \text{target}, & \text{when } e(t) > 0, \\ \text{target} - \min(\text{signal}(t)), & \text{when } e(t) < 0. \end{cases} \tag{4}$$

The decay ratio Δ is the value between two consecutive maxima of the controller output for a step change in the set point. This ratio is shown as

$$\Delta = \frac{c}{a}, \tag{5}$$

where the parameter a is the amplitude of the oscillations in the instant $t - 1$ and c the amplitude of the oscillation in the time t.

Let Γ be the settling time, that is, the time required for the controller output to stabilize at p% of the set point. According to literature, the recommended value of p is 0.02% [56, 57]:

$$\Gamma = \frac{t}{T_{\max}}, \quad \text{when } |\text{signal}(t) - \text{signal}(t-1)| < p, \tag{6}$$

where the parameter t is the amount of time in which the output signal is below p and $T_{\max}$ is the maximum value of the sampling time, considering the signal as the output value of the system.

The steady-state error Θ is the gap between the stabilized output signal of the system and the set point, after N controller iteration. In this paper, the selected value of N is 350, time enough for the controller to reach the set point (target):

$$\Theta = |\text{signal}(N) - \text{target}|. \tag{7}$$

Figure 4 shows the desired (blue) signal and the estimated/target (red) signal using the parameters computed by a particle swarm optimization algorithm: $(K_p = 14.1300, K_i = 4.6007, K_d = 0.00100000)$ with fitness f_1 value of 48.47595, $(K_p = 10.899968, K_i = 51.6346372, K_d = 0.001000000)$ with fitness f_2 value of 103.0716, and $(K_p = 10.901181, K_i = 100.000\,0000, K_d = 0.001000000)$ with fitness f_3 value of 26.21848. It can be seen that, in Figure 4(b), the peaks are smaller and the changes in signal are not as abrupt as in Figure 4(a), but Figure 4(c) shows the best results if using the full fitness function with all terms (overshoot, decay, setting time, and steady-state error), which is the expected and desired behaviour of the PID controller. Figure 4(c) is equivalent to the classical computation of PID.

4. Results

The particle swarm optimization algorithm has shown good results, see Figure 4, combined with an objective function f_3.

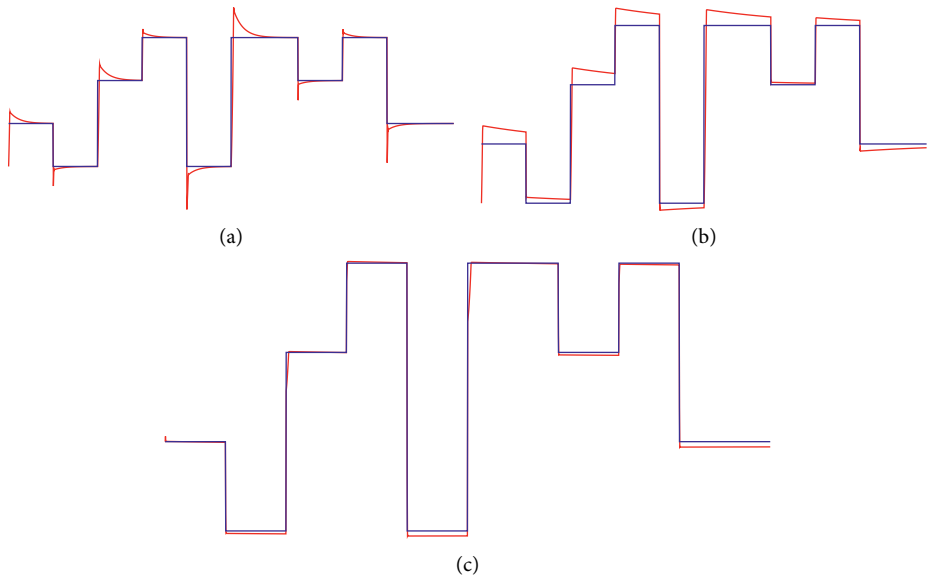

Figure 4: Optimization results using a PSO algorithm with 200 iterations and a population size of 20. Mean square error fitness f_1: (a) equation (1) vs. custom fitness f_2, (b) equation (2) vs. improved custom fitness f_3, and (c) equation (3). (a) Mean square error (equation (1)). (b) Overshoot and decay (equation (2)). (c) Overshoot, decay, setting time, and steady-state error as fitness function (equation (3)).

This section shows the results obtained using other metaheuristics (particle swarm optimization, whale optimization algorithm, differential evolution, clonal selection algorithm, cat swarm optimization, artificial bee colony algorithm, shuffled frog leaping, grey wolf optimizer, ant lion optimizer, firefly algorithm, dragonfly algorithm, harmony search algorithm, genetic algorithm, grasshopper optimization algorithm, sine cosine algorithm, moth flame optimizer, cuckoo search, bat algorithm, gravitational based search and black hole optimization, and krill herd algorithm) and the proposed three fitness functions f_1, f_2, and f_3. Optimization parameters have been set to the same values for all algorithms, 100 iterations, and a population size of 15 individuals with $(K_p, K_i, K_d) \in [0.001, 100]$, in order to get a similar testing environment for all algorithms.

The first approach consists in using the mean square error as a fitness function (see Table 5, equation (1), and Figure 5). In this case, some algorithms do not get a right solution since the output signal has a great oscillation pattern, but the ones that get good results produce peaks in the step function. This output is similar to the classical PID mathematical computation [49–51]. This behaviour could be considered a valid solution in other environments/problems but not in the case of a self-driving car with autonomous or remote speed controller as there are peaks and the car speed is not controlled in the transitions.

The second approach consists in using the overshoot and decay parameters (see Table 6, equation (2), and Figure 6). Now, the results are better than in the previous case since peaks disappear and the speed is controlled in a less abrupt mode (smooth transitions). As in the other case, some algorithms present oscillations, which is due to the lack of iterations or population size.

Finally, fitness function f_3, with the lineal combination of overshoot, decay, setting time, and steady-state error, is tested against the metaheuristics (see Table 7, equation (3), and Figure 7). The figure shows that the output signal fits the desired signal with no peaks and with an almost exact matching.

The main contribution of this paper is the use of the weighted fitness function definition f_3, as well as the use of this function as the basis to perform a metaheuristics optimization of the gains of a PID controller. The results of classical PID optimization usually generate controllers with an oscillating output. However, this oscillating output can be corrected using a low-pass filter, taking into account that the performance criteria always have less quality than metaheuristics optimization.

This paper has presented the results of metaheuristic optimization algorithms, based on different error metrics (1)–(3). These equations define the error as a lineal combination of the four estimators explained above. This error is calculated for a set point sequence instead of for a single set point; therefore, the controller obtained is more robust in new situations. In addition, the error is calculated over a different sequence than that used in the tuning, so the estimation of the controller error is more realistic.

Using these defined fitness functions and a randomly generated sequence of set points, a set of different gains has been obtained for the PID speed controller of an autonomous vehicle through a series of metaheuristics-based optimization techniques.

With most of the optimization algorithms studied, the ground truth error is improved. However, CS, CSO, and MFO perform worse than this ground truth. This fact could be due to the existence of multiple local minima in the parameter space for PID controller gains, which means that local search algorithms show premature convergence. In particular, the MFO algorithm finds different optima in some executions, seemingly indicating the existence of multiple local minima.

Table 5: Optimization solution values using proposed metaheuristics, using fitness f_1, see equation (1) and Figure 5.

Algorithm	K_p	K_i	K_d	Fitness
ABC	12.469341	4.48158890	0.00100000	48.50611
ACO	16.046610	35.68476668	0.00100000	51.17129
ALO	12.146504	4.21416994	0.00100000	48.49356
BA	5.093248	2.85260861	4.72422839	202.35233
BHO	1.734270	0.07296765	5.29040061	78.61255
CLONALG	32.463294	61.55373554	0.54027825	239.91586
CS	**14.432591**	**12.47369880**	**0.00100000**	**49.47551**
CSO	14.130092	4.60077827	0.00100000	48.47595
DA	14.130092	4.60078172	0.00100000	48.47595
DE	41.961478	1.82844085	20.91916782	490.12491
FFA	2.556885	1.04583741	0.55790335	71.08043
GA	17.332015	5.53366128	0.00100000	49.89310
GBS	27.031364	79.78306112	0.00100000	63.04356
GOA	14.132092	4.58780177	0.00100000	48.47596
HS	5.162984	1.35881040	1.38566380	84.41693
KH	17.009334	0.95704153	0.08686292	75.33240
MFO	14.130088	4.60078202	0.00100000	48.47595
PSO	26.413843	33.36398724	0.00100000	62.35269
SCA	14.209402	4.83498450	0.00100000	48.47908
SFL	5.972453	14.85325707	2.00940154	206.63336
WOA	14.128297	4.56045037	0.00100000	48.47605

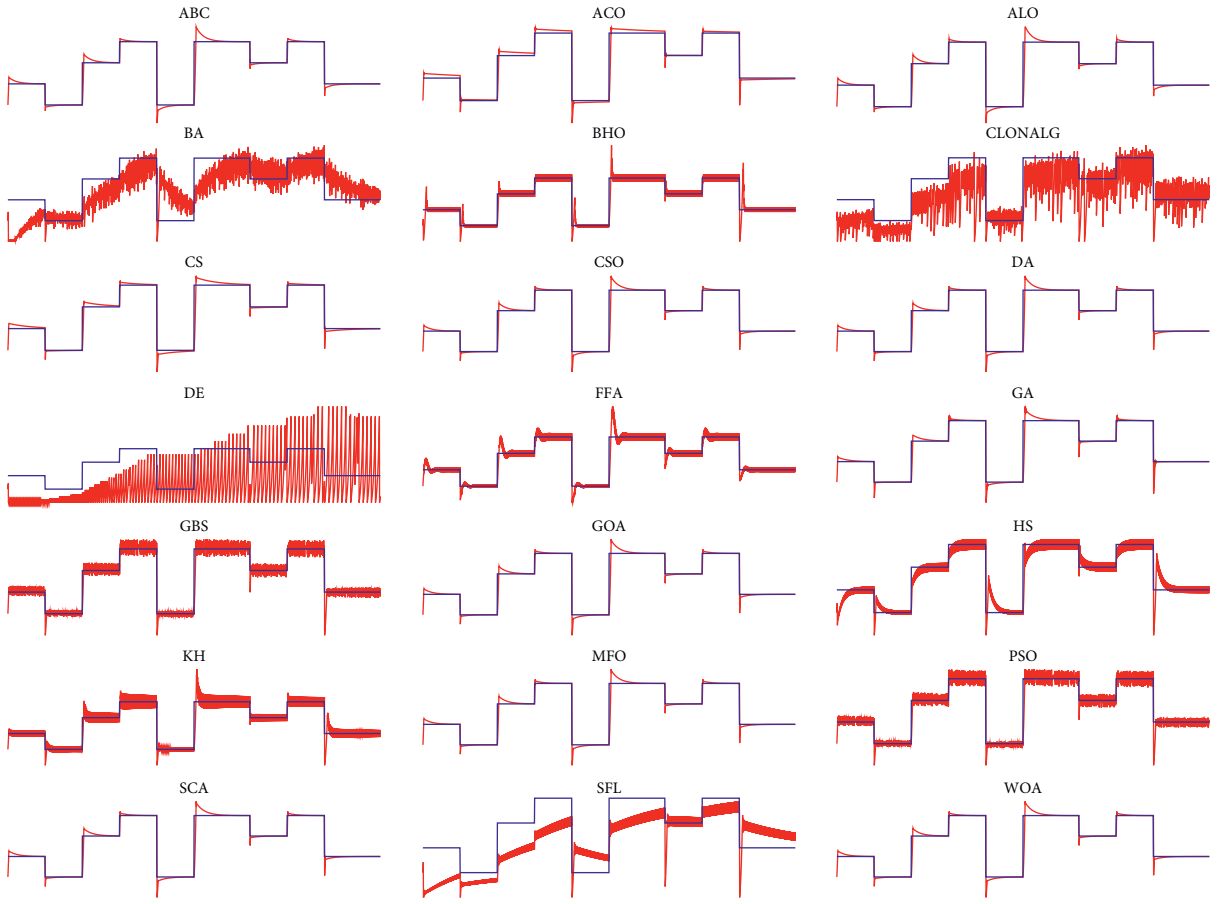

Figure 5: Results of 21 optimization metaheuristics using the mean square error as the fitness/objective function f_1: the red line is target/output signal and the blue line is desired signal, see equation (1) and Table 5. Parameters have been set to the same values for all algorithms, 100 iterations, and a population size of 15 individuals with $(K_p, K_i, K_d) \in [0.001, 100]$, in order to get a similar testing environment for all algorithms.

Table 6: Optimization solution values using proposed metaheuristics, using fitness f_2, see equation (2) and Figure 6.

Algorithm	K_p	K_i	K_d	Fitness
ABC	1.247355	29.8497584	2.024345564	127.5915
ACO	10.900608	64.1849888	0.001000000	103.1267
ALO	10.030650	13.1977015	0.001000000	108.4750
BA	60.898153	24.6557423	3.061338914	169.5129
BHO	6.748417	51.3241341	0.068173507	111.4488
CLONALG	4.201685	55.5608255	0.011870125	127.6016
CS	10.863056	13.4336943	0.001000000	107.0472
CSO	10.901259	100.0000000	0.001000000	103.5151
DA	10.900160	51.8360956	0.001000000	103.0721
DE	47.586536	34.9134975	3.302378887	183.9771
FFA	2.910704	35.4693957	1.001861920	129.7192
GA	10.847629	34.3291956	0.001000000	103.3316
GBS	0.417065	0.8933817	52.958178733	128.2626
GOA	10.899268	48.2122536	0.001000758	103.0773
HS	21.376061	1.3812947	49.333388968	181.5794
KH	60.249693	11.1017443	25.715143616	249.9684
MFO	**10.900022**	**51.4431324**	**0.001000000**	**103.0715**
PSO	10.899968	51.6346372	0.001000000	103.0716
SCA	10.876744	46.7826595	0.001032572	103.1077
SFL	75.568124	23.1263233	58.945069279	263.1667
WOA	10.823591	62.7439498	0.001796744	103.2075

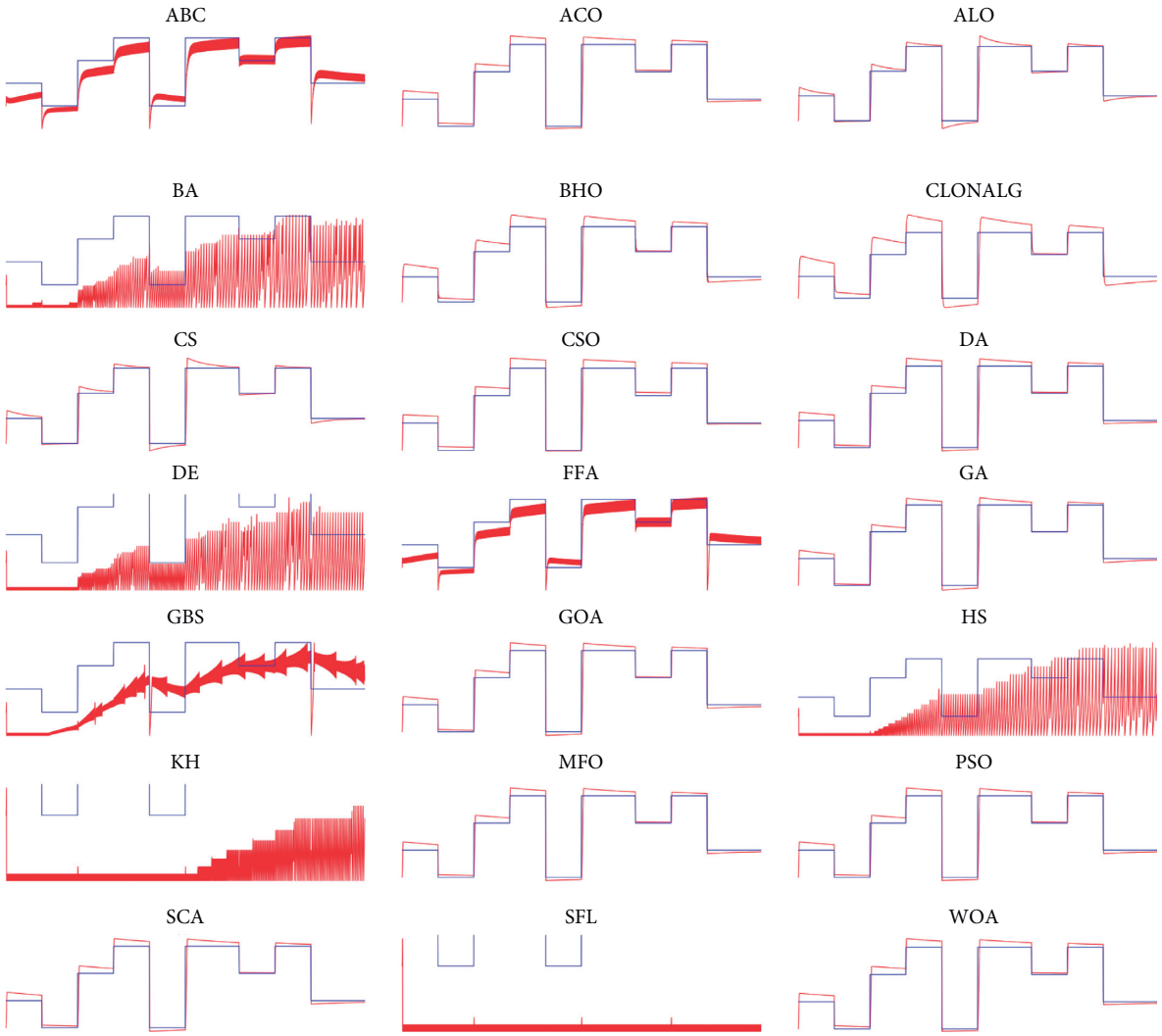

Figure 6: Results of 21 optimization metaheuristics using a fitness/objective function f_2 with overshoot and decay components: the red line is target/output signal and the blue line is desired signal, see equation (2) and Table 6. Parameters have been set to the same values for all algorithms, 100 iterations, and a population size of 15 individuals with $(K_p, K_i, K_d) \in [0.001, 100]$, in order to get a similar testing environment for all algorithms.

Table 7: Optimization solution values using proposed metaheuristics, using fitness f_3, see equation (3) and Figure 7.

Algorithm	K_p	K_i	K_d	Fitness
ABC	4.317014	100.0000000	0.414558038	38.13464
ACO	10.899466	44.7705683	0.001000000	27.30494
ALO	10.465441	29.5566926	0.001000000	28.41234
BA	1.380994	10.2406319	2.940162854	50.77700
BHO	1.215425	99.6389347	1.649872970	38.02265
CLONALG	16.254893	88.6727789	0.319180483	51.81580
CS	9.001000	9.0010000	0.001000000	33.31014
CSO	**10.901176**	**100.0000000**	**0.001000000**	**26.21848**
DA	2.181195	16.8407815	0.924729639	37.82754
DE	76.884508	65.8029157	0.692211458	74.54958
FFA	9.095726	99.9275577	0.019747615	27.25296
GA	100.000000	27.9241829	1.851140432	77.36712
GBS	32.971795	68.7382582	0.001000000	44.95544
GOA	10.801275	100.0000000	0.001930287	26.26413
HS	6.971770	3.4059226	1.211073046	45.29076
KH	8.067583	12.4642292	2.274522387	52.22566
MFO	**10.901181**	**100.0000000**	**0.001000000**	**26.21848**
PSO	27.551768	25.0129883	35.913471627	134.51197
SCA	10.775543	100.0000000	0.001000000	26.27785
SFL	2.687796	3.1262379	1.112656001	39.43179
WOA	61.659377	0.6926213	76.733240892	84.92107

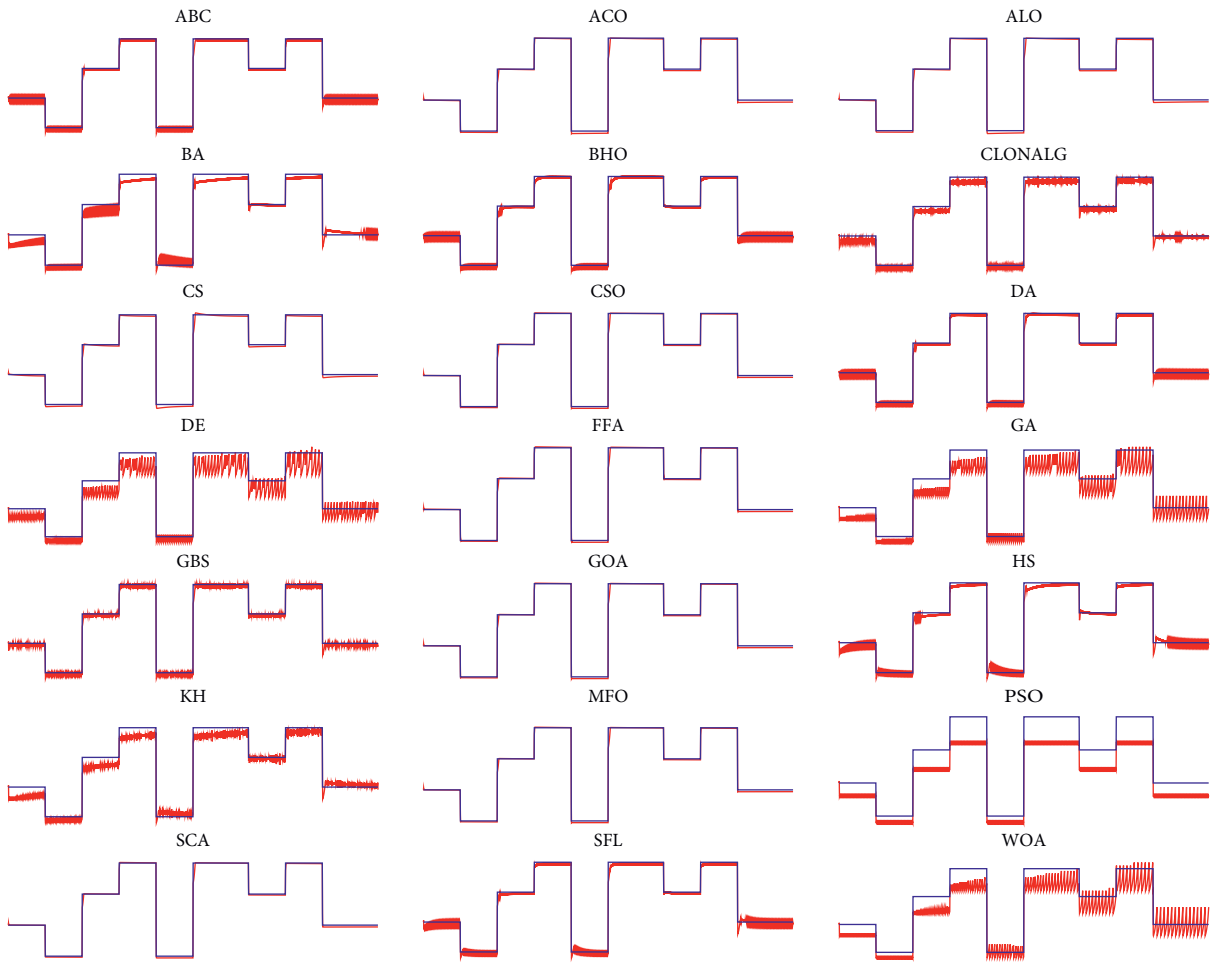

Figure 7: Results of 21 optimization models using a fitness/objective function f_3 with overshoot, decay, setting time, and steady-state error components: the red line is target/output signal and the blue line is desired signal, see equation (3) and Table 7. Parameters have been set to the same values for all algorithms, 100 iterations, and a population size of 15 individuals with $(K_p, K_i, K_d) \in [0.001, 100]$, in order to get a similar testing environment for all algorithms.

5. Conclusions

This paper shows the performance of the computation power of several evolutionary algorithms to optimize the gains (K_p, K_i, K_d) of the speed PID controller of a self-driving vehicle. Three fitness functions have been used to optimize the controller gains. Those functions are based in a combination of classic error metrics of the controller: overshoot, decay ratio, settling time, and steady-state error. The results of the different optimizations of the PID speed controller gains have been compared with each other, both for the sequence of step set points uses as the target of the optimization and for a totally new randomly generated sequence, unknown by the system. This system has been implemented by means of a dynamic model of the vehicle in simulation that represents the speed behaviour in a realistic way.

The fitness function proposed to measure the quality of a PID controller according to the traditional PID quality metrics reported by the literature, meaning that the minimization of this fitness implies the joint optimization of these metrics, since some can imply disturbances in the rest. The weight of each of them has been carefully tuned to find an optimal overall solution.

Except for the methods used in the paper, some of the most representative computational intelligence algorithms can be used to solve the problem, such as earthworm optimization algorithm (EWA) [40], monarch butterfly optimization (MBO) [41, 42], moth search (MS) algorithm [43], elephant herding optimization (EHO) [45], Harris hawks optimization (HHO) [46], and slime mould algorithm (SMA) [44].

6. Future Research

As possible future research lines, coefficients in equation (3) could be tuned using different algorithms of literature, analysing the optimization rate compared to the methods presented in this paper. The results of this paper have been tested in simulation; as future work, we propose to test the optimized controllers in real physical autonomous vehicles as well as to implement an online optimization using the best techniques studied in this paper.

Some dynamic metaheuristic optimization models, such as multilayer neural networks (MLP), could be implemented using the particular structure of MLPs. The MLP benefits from the complex architecture of the neural networks and its transformations in order to achieve new optimal solutions. In terms of convergence, the relationship between random exploitation and each parameter under asymmetric interval is derived and an iterative convergence of neural networks is proved mathematically [60]. Other optimization techniques such as the water cycle algorithm could also be applied to solve the problem proposed [61].

Abbreviations

ABC: Artificial bee colony algorithm
ALO: Ant lion optimizer
BA: Bat algorithm
BHO: Black hole optimization
CLONALG: Clonal selection algorithm
CS: Cuckoo search
CSO: Cat swarm optimization
DA: Dragonfly algorithm
DE: Differential evolution
EHO: Elephant herding optimization
EWA: Earthworm optimization algorithm
FFA: Firefly algorithm
GA: Genetic algorithms
GBS: Gravitational-based search
GOA: Grasshopper optimisation algorithm
GWO: Grey wolf optimizer
HHO: Harris hawks optimization
HS: Harmony search algorithm
KH: Krill herd algorithm
LiDAR: Light detection and ranging sensor
MBO: Monarch butterfly optimization
MFO: Moth flame optimizer
MS: Moth search algorithm
NNA: Neural network algorithm
NP: Nondeterministic polynomial-time
PID: Proportional integrative derivative
PSO: Particle swarm optimization
SCA: Sine cosine algorithm
SFL: Shuffled frog leaping
SMA: Slime mould algorithm
SSA: Salp swarm algorithm
WCA: Water cycle algorithm
WOA: Whale optimization algorithm.

Data Availability

No data is available.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

J.E.N.H. and N.G.B. conceptualized the study; L.F.d.M.L. and F.S. helped with software; J.E.N.H., F.S. and L.F.d.M.L. investigated the study; N.G.B. supervised; all authors analysed the data and wrote the paper. Luis Fernando de Mingo López, Francisco Serradilla, Jose Eugenio Naranjo Hernández, and Nuria Gómez Blas authors contributed equally to this work.

Acknowledgments

This work was partially supported by the Comunidad de Madrid under Convenio Plurianual with the Universidad Politécnica de Madrid in the actuation line of Programa de Excelencia para el Profesorado Universitario. This work has been partially supported by projects Seguridad de Vehículos AUTOmóviles para un TRansporte Inteligente, Eficiente y Seguro (SEGVAUTO4.0 P2018/EMT-4362) and CICYT (PID2019-104793RB-C33).

References

[1] S. Nesmachnow, "An overview of metaheuristics: accurate and efficient methods for optimisation," *International Journal of Metaheuristics*, vol. 3, p. 320, 2014.

[2] Z. Beheshti and S. M. Shamsuddin, "A review of population-based meta-heuristic algorithm," *International Journal of Advances in Soft Computing and its Applications*, vol. 5, pp. 1–35, 2013.

[3] O. P. Goldreich, *NP-Completeness: The Basics of Computational Complexity*, Cambridge University Press, Cambridge, MA, USA, 1st edition, 2010.

[4] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The Arithmetic optimization algorithm," *Computer Methods in Applied Mechanics and Engineering*, vol. 376, Article ID 113609, 2021.

[5] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer Publishing Company, New York, NY, USA, 1st edition, 2007.

[6] L. Nunes de Castro, "Fundamentals of natural computing: an overview," *Physics of Life Reviews*, vol. 4, pp. 1–36, 2007.

[7] Z. J. Lee, S. F. Su, C. C. Chuang, and K. H. Liu, "Genetic algorithm with ant colony optimization (GA-ACO) for multiple sequence alignment," *Applied Soft Computing*, vol. 8, pp. 55–78, 2008.

[8] M. Dorigo and T. Stützle, *Ant Colony Optimization*, Bradford Company, Scituate, MA, USA, 2004.

[9] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, Oxford, UK, 1999.

[10] M. Dorigo, V. Maniezzo, and A. Colorni, "The ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26, pp. 29–41, 1996.

[11] A. Askarzadeh, "Bird mating optimizer: an optimization algorithm inspired by bird mating strategies," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, pp. 1213–1228, 2014.

[12] E. Duman, M. Uysal, and A. F. Alkaya, "Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem," *Information Sciences*, vol. 217, pp. 65–77, 2012.

[13] A. Sharma, "A new optimizing algorithm using reincarnation concept," in *Proceedings of the 2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 281–288, Budapest, Hungary, November 2010.

[14] H. Nguyen and B. Bhanu, "Zombie survival optimization: a swarm intelligence algorithm inspired by zombie foraging," in *Proceedings of the 2012 21st International Conference on Pattern Recognition (ICPR 2012)*, pp. 987–990, IEEE Computer Society, Los Alamitos, CA, USA, June 2012.

[15] D. Teodorovic, P. Lucic, G. Markovic, and M. D. Orco, "Bee colony optimization: principles and applications," in *Proceedings of the 2006 8th Seminar on Neural Network Applications in Electrical Engineering*, pp. 151–156, Serbia and Montenegro, Belgrade, September 2006.

[16] H. J. Sung, "Queen-bee evolution for genetic algorithms," *Electronics Letters*, vol. 39, pp. 575-576, 2003.

[17] H. A. Abbass, "MBO: marriage in honey bees optimization-a Haplometrosis polygynous swarming approach," *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, pp. 207–214, 2001.

[18] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proceedings-IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[19] L. N. de Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 239–251, 2002.

[20] S. Mirjalili and A. Lewis, "The Whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[21] M. Eusuff, K. Lansey, and F. Pasha, "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization," *Engineering Optimization*, vol. 38, pp. 129–154, 2006.

[22] T. Jayabarathi, S. Chalasani, Z. A. Shaik, and N. D. Kodali, "Hybrid differential evolution and particle swarm optimization based solutions to short term hydro thermal scheduling," *WSEAS Transactions on Power Systems*, vol. 11, pp. 245–254, 2007.

[23] M. Pant, R. Thangara, C. Grosan, and A. Abraham, "Hybrid differential evolution-particle swarm optimization algorithm for solving global optimization problems," in *Proceedings of the Third IEEE International Conference on Digital Information Management*, pp. 18–24, London, UK, November 2008.

[24] S. C. Chu, P. w. Tsai, and J. S. Pan, *Cat Swarm Optimization. PRICAI 2006: Trends in Artificial Intelligence*, Q. Yang and G. Webb, Eds., Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[25] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[26] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in *Proceedings of the 12th International Fuzzy Systems Association World Congress on Foundations of Fuzzy Logic and Soft Computing*, Cancun, Mexico, June 2007.

[27] X. S. Yang, *Firefly Algorithms for Multimodal Optimization. Stochastic Algorithms: Foundations and Applications*, O. Watanabe and T. Zeugmann, Eds., Springer Berlin Heidelberg, Berlin, Heidelberg, Germany, 2009.

[28] S. Mirjalili, S. M. Mirjalili, and A. G. W. Lewis Optimizer, *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[29] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, pp. 1053–1073, 2016.

[30] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: theory and application," *Advances in Engineering Software*, vol. 105, pp. 30–47, 2017.

[31] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, Cambridge, MA, USA, 1992.

[32] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, pp. 1567–1579, 2007.

[33] S. S. Mirjalili, "CA: a Sine Cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016.

[34] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[35] A. H. Gandomi and A. H. K. Alavi herd, "A new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, pp. 4831–4845, 2012.

[36] Y. Xin-She and H. G. Amir, "Bat algorithm: a novel approach for global engineering optimization," *Engineering Computations*, vol. 29, pp. 464–483, 2012.

[37] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, pp. 2232–2248, 2009.

[38] X. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the 2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pp. 210–214, Coimbatore, India, December 2009.

[39] A. Hatamlou, "Black hole: a new heuristic optimization approach for data clustering," *Information Sciences*, vol. 222, pp. 175–184, 2013.

[40] G. Wang, S. Deb, and L. Coelho, "Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems," *International Journal of Bio-Inspired Computation*, vol. 12, pp. 1–22, 2018.

[41] H. Faris, I. Aljarah, and S. Mirjalili, "Improved Monarch butterfly optimization for unconstrained global search and neural network training," *Applied Intelligence*, vol. 48, pp. 445–464, 2018.

[42] Y. Feng, S. Deb, G. Wang, and A. H. Alavi, "Monarch butterfly optimization: a comprehensive review," *Expert Systems with Applications*, vol. 168, Article ID 114418, 2021.

[43] G. G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, pp. 151–164, 2016.

[44] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.

[45] G. Wang, S. Deb, and d. S. Coelho, "Elephant herding optimization," in *Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, pp. 1–5, Bali, Indonesia, December 2015.

[46] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[47] L. Abualigah and A. Diabat, "A comprehensive survey of the Grasshopper optimization algorithm: results, variants, and applications," *Neural Computing and Applications*, vol. 32, pp. 15533–15556, 2020.

[48] L. Abualigah, "Group search optimizer: a nature-inspired meta-heuristic optimization algorithm with its results, variants, and applications," *Neural Computing and Applications*, vol. 33, pp. 2949–2972, 2020.

[49] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Journal of Dynamic Systems, Measurement, and Control*, vol. 115, pp. 220–222, 1993.

[50] K. Ogata, *Modern Control Engineering*, Prentice Hall PTR, Hoboken, NJ, USA, 4th edition, 2001.

[51] K. L. Chien, J. A. Hrons, and J. B. Reswick, "On the automatic control of generalized passive systems," *Transactions of the American Society of Mechanical Engineeing*, vol. 74, pp. 175–185, 1972.

[52] G. H. Cohen, "Theoretical consideration of retarded control," *Transactions of the American Society of Mechanical Engineeing*, vol. 75, pp. 827–834, 1953.

[53] L. Mora, R. Lugo, C. Moreno, and J. E. Amaya, "Parameters optimization of PID controllers using metaheuristics with physical implementation," in *Proceedings of the 2016 35th International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1–8, Valparaíso, Chile, October 2016.

[54] S. K. Valluru and M. Singh, "Optimization strategy of bio-inspired metaheuristic algorithms tuned PID controller for PMBDC actuated robotic manipulator," *Procedia Computer Science*, vol. 171, pp. 2040–2049, 2020.

[55] R. Matušů, K. Janprom, W. Permpoonsinsup, and S. Wangnipparnto, "Intelligent tuning of PID using metaheuristic optimization for temperature and relative humidity control of comfortable rooms," *Journal of Control Science and Engineering*, vol. 2020, Article ID 2596549, 2020.

[56] F. Serradilla, N. Cañas, and J. E. Naranjo, "Optimization of the energy consumption of electric motors through metaheuristics and PID controllers," *Electronics*, vol. 9, p. 1842, 2020.

[57] J. E. Naranjo, F. Serradilla, and F. Nashashibi, "Speed control optimization for autonomous vehicles with metaheuristics," *Electronics*, vol. 9, p. 551, 2020.

[58] J. Bhookya and R. K. Jatoth, "Optimal FOPID/PID controller parameters tuning for the AVR system based on sine-cosine-algorithm," *Evolutionary Intelligence*, vol. 12, pp. 725–733, 2019.

[59] S. Zhang and L. Liu, "Normalized robust FOPID controller regulation based on small gain theorem," *Complexity*, vol. 2018, Article ID 5690630, 2018.

[60] A. Sadollah, H. Sayyaadi, and A. Yadav, "A dynamic metaheuristic optimization model inspired by biological nervous systems: neural network algorithm," *Applied Soft Computing*, vol. 71, pp. 747–782, 2018.

[61] H. Eskandar, A. Sadollah, A. Bahreininejad, and M. Hamdi, "Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems," *Computers and Structures*, vol. 110-111, pp. 151–166, 2012.