Article

Metaheuristics-Based Optimization of a Robust GAPID Adaptive Control Applied to a DC Motor-Driven Rotating Beam with Variable Load

Fábio Galvão Borges 1, Márcio Guerreiro 2, Paulo Eduardo Sampaio Monteiro 1, Frederic Conrad Janzen 3, Fernanda Cristina Corrêa 1, Sergio Luiz Stevan, Jr. 1, Hugo Valadares Siqueira 1,2,3,* and Mauricio dos Santos Kaster 1,3

1 Graduate Program in Electrical Engineering (PPGEE), Federal University of Technology—Paraná (UTFPR), R. Dr. Washington Subtil Chueire, 330, Jardim Carvalho, Ponta Grossa 84017-220, PR, Brazil
2 Graduate Program in Industrial Engineering (PPGEP), Federal University of Technology—Paraná (UTFPR), R. Dr. Washington Subtil Chueire, 330, Jardim Carvalho, Ponta Grossa 84017-220, PR, Brazil
3 Electrical Engineering Department, Federal University of Technology—Paraná (UTFPR), R. Dr. Washington Subtil Chueire, 330, Jardim Carvalho, Ponta Grossa 84017-220, PR, Brazil

* Correspondence: hugosiqueira@utfpr.edu.br; Tel.: +55-42-3220-4825

Abstract: This work aims to analyze two metaheuristics optimization techniques, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), with six variations each, and compare them regarding their convergence, quality, and dispersion of solutions. The optimization target is the Gaussian Adaptive PID control (GAPID) to find the best parameters to achieve enhanced performance and robustness to load variations related to the traditional PID. The adaptive rule of GAPID is based on a Gaussian function that has as adjustment parameters its concavity and the lower and upper bound of the gains. It is a smooth function with smooth derivatives. As a result, it helps avoid problems related to abrupt increases transition, commonly found in other adaptive methods. Because there is no mathematical methodology to set these parameters, this work used bio-inspired optimization algorithms. The test plant is a DC motor with a beam with a variable load. Results obtained by load and gain sweep tests prove the GAPID presents fast responses with very low overshoot and good robustness to load changes, with minimal variations, which is impossible to achieve when using the linear PID.

Keywords: GAPID control; bio-inspired metaheuristics optimization; genetic algorithm; particle swarm optimization

1. Introduction

Currently, most industrial processes still employ traditional controllers such as Proportional–Integral–Derivative (PID), which are simple, practical, have well-established design techniques, and come pre-packaged in most control devices, such as programmable logical controllers (PLCs) [1]. However, linear controllers have limited performance due to their intrinsic linearity [2–4]. Nonlinear controllers, in most cases, can easily overcome such limitations [5]. Adaptive control is a class of nonlinear control that takes the linear controller structure and adapts the control parameters dynamically, following a special rule that defines these parameters upon some operating conditions [6]. Other techniques use adaptive practices to adjust functional parameters, such as Fuzzy Adaptive [7,8] and Neural Adaptive [9–12] control techniques.

Several works propose adaptive PID-based techniques that enhance the controller performance, taking advantage of the robust structure of the traditional PID. One option employs Fuzzy logic to adapt the PID gains. In [13], the membership functions are described for inputs (error and derivative) and outputs ($K_p$, $K_i$, and $K_d$ gains); as such, these gains can adapt according to the designed membership functions. The advantage of using Fuzzy...
techniques relies on requiring only basic knowledge of how the adaptive gains are expected to perform in the controller. The disadvantage is that the resulting controller is likely not optimal.

Another option consists of building adaptation functions based on the error signal to adapt the PID gains dynamically. In [14], a nonlinear PID (NL-PID) is compared to a PID tuned by a Genetic Algorithm (GA) and a PID adjusted by the Ziegler–Nichols method. This NL-PID takes a Gaussian function of the error and employs linear functions as the adaptive rules for the gains. Such a procedure demands relatively low computing power, but the method to define the adaptive parameters of the NL-PID are not straightforward, and it is unlikely to lead to an optimal controller.

In [15], a nonlinear adaptive PID with parameters defined by Genetic Algorithm (GA) was described. The adaptive PID rules were determined by 11 parameters, representing a complex problem optimized by the GA. Furthermore, its operation needs to calculate five exponential functions for each control iteration which takes a considerable computational power.

The Gaussian Adaptive PID (GAPID) is a kind of adaptive control firstly proposed by this research group in 2015 [16], where the adaptive rule follows a Gaussian function of the system error. A Gaussian curve allows for transition from low to high gain in a smooth fashion. It demands light computational effort by using only one exponential function in each iteration, considering that more specific functions save lots of computing power in massive calculations.

The GAPID control technique currently has no mathematical design procedure. The problem of parameter tuning belongs to the NP-hard challenges, which cannot be addressed with traditional deterministic methods. An alternative to overcome these problems is to use bio-inspired optimization metaheuristics that allow for finding a better controller and eventually optimum parameters in a reasonable period [1]. Such an optimization task poses a particular difficulty because the problem is multi-modal, where distinct solutions have very similar performance values. Furthermore, several solutions are not robust because the system performs well only with the load defined for optimization and decreases considerably with other load values [17].

Regarding bio-inspired metaheuristics, the inspiration for the development of such methods lies, for example, in the Laws of the Darwinian Evolution (evolutionary algorithms) or the collective intelligent behavior of groups of animals (swarm-inspired methods) [18–20]. These algorithms are endowed with an intrinsic capability of finding local or, eventually, the globally optimum values of the coefficients of a system.

In this sense, intelligent algorithms have been successfully employed to solve multi-modal optimization tasks from various real-world problems. For example, in [21], the authors proposed the use of an improved version of the Salp Swarm Algorithm to deal with feature selection problems; variations of the Moth-flame algorithm, Particle Swarm Optimization (PSO), Genetic Algorithm (GA), and Bees Life Algorithm (BLA) were addressed for cyber-physical system applications in real fog computing; a hybrid proposition among Firefly Algorithm and PSO is proposed in [22] for optimal cluster head selection in wireless sensor networks with the goal of increasing the energy efficiency of the system; notwithstanding, nonlinear control is the target of many studies involving bio-inspired optimization, such as [23–28].

In [17,29] some applications of GAPID were explored by using bio-inspired metaheuristics optimization (Genetic Algorithm and Particle Swarm Optimization (PSO)) to find optimal parameters for the GAPID. In [30] other metaheuristics (Whale Optimization Algorithm (WOA) and Artificial Bee Colony (ABC) targeted to GAPID were also studied, using PSO as a common comparison choice to verify different metaheuristics algorithm effectiveness. This study demonstrated that WOA achieved a slightly better optimization result, but PSO converges faster and presented the smallest dispersion of the final results.

Considering the assumption mentioned above, this work aims to analyze the performances of a set of optimization algorithms in charge of finding optimal solutions for the spe-
cial case of the GAPID controller. However, differently from the previous works [1,17,29], which applied optimization algorithms to GAPID for the plant with fixed load, the novelty here is that variable load is now being considered in the optimization. This demanded some modifications in the optimization process to result in a more robust controller; in this case, for each iteration and each individual, several loads within the load range were tested, and the mean of the fitness for each load was taken as the individual fitness in that iteration. In practice, such a procedure is capable of leading to a robust controller that is expected to perform optimally within a predefined load range.

To this end, six variations of the Genetic Algorithm and six variations of the Particle Swarm Optimization were selected and applied to a DC motor-driven rotating beam [31–35]. In the first moment, the different optimization strategies to obtain the optimal parameters of the GAPID were examined, then analyzed and compared regarding their efficiencies. Afterward, their performances in an experimental test plant were verified.

The GAPID tuned by optimization algorithms is a robust control technique sharing parameters with a previously designed PID, which facilitates its adoption in the industry but also offers some limitations. In most cases, it has the same restrictions as the PID, such as the windup problem in the integral component and noise amplification in the derived component, which is mitigated in the GAPID for minor errors. The other issue is that the plant must be stressed by a series of tests to obtain the fitness values for each individual in each iteration. It is impractical to perform such a task in a real-world plant, so an accurate model must be derived and the heavy lifting transferred to a computer simulation environment.

The remainder of the paper is divided as follows: Section 2 presents the subjects under study, which are the GAPID control and the optimization algorithms used to find optimal parameters for the GAPID. Section 3 presents the test plant model. Section 4 discusses the methodology employed for simulations and experimental analysis. Section 5 presents the results obtained from simulation, comparisons, and comments about the optimization algorithms and the experimental results obtained in the physical plant. Finally, Section 6 presents the main conclusions and future perspectives.

2. Theoretical Foundation

In this section, the GAPID controller technique and Bio-inspired Optimization algorithms that will be utilized in the work are presented, mainly the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The Fitness function is also covered, which is very important because it directs the optimization toward the solution. This information is the key to addressing the variations applied in the Optimization algorithm, stated in Section 4, for posterior performance analysis.

2.1. GAPID Controller

It is known that nonlinear controllers are more efficient than linear controllers; the problem is that they are harder to design. This work employs the adaptive PID controller, where the proportional, integral, and derivative gains adapt according to a Gaussian function of the input error $\varepsilon$. The conceptual idea is to transition between two gain levels, $k_0$ and $k_1$, using a smooth process with smooth derivatives as the adaptation rule.

Smooth functions are desirable because they avoid unexpected behavior caused by discontinuous gain transitions. The proposed function is Gaussian-like, as stated in Equation (1)

$$\lambda(\varepsilon) = k_1 - (k_1 - k_0)e^{-\delta\varepsilon^2}$$

(1)

where $\varepsilon$ is the input error, $k_0$ is the gain when $|\varepsilon| = 0$, $k_1$ is the gain when $|\varepsilon| = \infty$, and $\delta$ is the openness degree of the Gaussian function. This function will describe the adaptive PID gains $\lambda_p(\varepsilon)$, $\lambda_i(\varepsilon)$, and $\lambda_d(\varepsilon)$ [1].

As one can see, this function has three parameters: $k_0$, $k_1$, and $\delta$. So, the design process is to define the correct values for these parameters. Figure 1a,b show the function shapes for $k_0 > k_1$, and $k_1 > k_0$, and Figure 1c shows the adjustable concavity degree.
As no mathematical design methodology has yet been developed to find proper parameter values for the GAPID, search algorithms are a good option. In this sense, this work used meta-heuristics optimization to find an optimal solution. The problem has two approaches: the free parameters and the linked parameter cases.

2.1.1. Free Parameters Case

The three PID gains turned adaptive by Gaussian functions become a nine-parameter design ($k_0, k_1,$ and $\delta$ of Equation (1) for each of the three gains functions), representing a complex problem to solve. Furthermore, there is the multi-modal problem, where several solutions may result in a near-best controller.

However, this problem can be reduced to one parameter: it is reasonable to state the derivative gain as zero when $\varepsilon = 0$, turning the PID into a PI when the output matches the input reference, helping to avoid the problem that noise provokes in the control signal due to the derivative component of PID controllers. Then, the overall problem becomes an eight-parameter design referred to as the free parameters case.

2.1.2. Linked Parameters Case

Considering that most industrial plants utilize PID controllers, several of which are hardly tuned in over long experimentation, switching to a nonlinear adaptive controller with non-tested parameters in a production plant is not easily accepted. In this sense, the proposal of setting an adaptive controller based on the already-determined PID gains is more plausible. The idea is to derive the Gaussian limits $k_0$ and $k_1$ as functions of the same parameter $x$ in the form $k_0 = x \cdot K_0$ and $k_1 = 1/x \cdot K_1$, where $K$ is the previously defined linear gain of the original PID. The derived Gaussian functions become

$$\lambda_p(\varepsilon) = x K_p - \left( x K_p - 1/x K_p \right) e^{-\delta_p \varepsilon^2}$$

$$\lambda_i(\varepsilon) = y K_i - \left( y K_i - 1/y K_i \right) e^{-\delta_i \varepsilon^2}$$

$$\lambda_d(\varepsilon) = z K_d e^{-\delta_d \varepsilon^2}$$

where $K_p, K_i,$ and $K_d$ are the PID gains. Equations (2)–(4) represent a six-parameter problem design ($x, y, z, \delta_p, \delta_i, \delta_d$), which will be referred as the linked parameters case.

As stated in [29], in most cases, the free parameters case leads to a better solution, usually representing a particular best solution that performs very well, but only with specific design conditions. If conditions change, the performance often decreases significantly. On the other hand, the case of the linked parameters, besides performing a little behind the case of the free parameters, is more tolerant to changes and exhibits better robustness to load variations. It also presents the additional advantage of being tied to the original gains that most controllers have already defined in their plants.

2.2. Bio-Inspired Optimization

Where finding an optimal solution may be impossible or impractical, heuristic techniques can speed up the process of finding a good one [36].
A meta-heuristic is defined, in computer science, as a high-level heuristic designed to find, generate, or select a search algorithm that could provide some satisfactory possible solutions [37].

In this work, two methods for optimization were used, the Genetic Algorithm (GA) and the Particle Swarm Optimization (PSO). For each of the methods, a set of six variations were examined.

2.2.1. Genetic Algorithm

The Genetic Algorithm (GA) is the first introduced bio-inspired meta-heuristic to deal with optimization problems [38]. Still, it is an essential evolutionary algorithm applied in many areas, such as clustering, time series forecasting, and control [1,17,39].

The GA was inspired by Darwinian evolution by natural selection [40]. Thus, the method is initialized, creating a population of candidate solutions (named individuals or chromosomes) that are randomly generated. Such agents are characterized by their coordinates on the problem-solving space, which means that an individual is a vector containing the values of the addressed problem parameters [41].

Each agent also receives a fitness, that is, a score representing the response quality leading to the system of interest. Bringing this idea to natural selection, solutions with higher fitness values tend to survive the selection process and maintain their genotype in the next generation. After the age of the population and fitness assignment, the first versions of the GA follow three basic steps: selection, crossover, and mutation.

The selection defines the individuals who pass through the other operations. Generally, all methods choose the best-ranked agents, the roulette wheel, and tournament [42]. In the first step, a roulette is created, in which each slice corresponds to an individual. However, the size of such a slice is proportional to the fitness, so that the best-ranked individuals present a higher probability of being selected. It is possible to choose the same agent more than once.

The tournament procedure is initiated by randomly selecting two individuals. Then, they compete in a way that the highest fitness is chosen. In this case, the selective pressure is lower than the roulette wheel [40].

After that, the crossover is applied between the two individuals of the population (parents) selected in the previous step. Such parents change some values of their genes, generating two new individuals (offspring), which replace the parents in the current population. The procedure is repeated until the population presents the same number as the original population. The most known method is the one-point crossover. The crossover is an exploitation procedure or a way to perform local search [38].

The last step is mutation. Usually, it presents a small probability of occurrence and is applied to all the genes of the entire population. Therefore, some may be changed, perturbing their values according to predefined distribution. Another possibility is to disturb the genes considering a predefined number of them. In this case, the probability of occurrence is 100% for the genes selected. This step allows for a global search to find some unexplored regions of the cost function [41]. After that, a new generation is obtained that passes through the same operations until the stop criterion. Figure 2a presents the steps of the traditional GA approach.

Another variation of the GA is based on using a sub-population formed by the sum of parents and offspring during a generation. In this case, the three steps mentioned above occur differently: crossover, mutation, and selection. This approach performs the creation of offspring after the crossover, but the parents remain, doubling the number of individuals in the population. After, the mutation is applied considering the entire population. Finally, the selection (roulette wheel or tournament, for example) is used to select the final population, considering a competition among parents and offspring. Figure 2b presents the steps of the GA-modified strategy.
Figure 2. Flowcharts of GA strategies: (a) traditional strategy; (b) modified strategy.

2.2.2. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a swarm-based computing technique developed by Kennedy and Eberhart [43], which was inspired by the social behavior of groups of animals, such as flocks of birds and schools of fishes.

PSO is a simple yet powerful optimization tool. Its main characteristic is simple agents working together, which can arise sufficient collective intelligence to conduct the operation, respecting simple rules [44]. Each particle has a position and velocity in the search space. These two variables respect the following rules in each iteration (Equations (5) and (6)).

\[
v_{i}^{n+1} = \omega v_{i}^{n} + r_{1}c_{1}(p_{best} - x_{i}^{n}) + r_{2}c_{2}(g_{best} - x_{i}^{n})
\]

\[
x_{i}^{n+1} = x_{i}^{n} + v_{i}^{n+1}
\]

where \(x_{i}^{n}\) and \(v_{i}^{n}\) are position and speed of the \(i\)-th particle in the \(n\)-th iteration, \(w\) is the inertial weight, \(r_{1}\) and \(r_{2}\) are random numbers between 0 and 1, \(c_{1}\) is the personal acceleration factor, \(c_{2}\) is the social acceleration factor, \(p_{best}\) is the best personal position achieved so far, and \(g_{best}\) is the best global position until the current iteration.

Figure 3 presents the steps of the PSO addressed.
Two main topologies are addressed in this work: (a) Global Topology: each particle considers as global best (g_{best}) the best-positioned particle for the sake of the entire population; (b) Ring Topology: the particle considers as global best (g_{best}) the best-positioned particle among its nearest topological neighbors, which are defined previously.

2.2.3. Fitness Function

The fitness function (FF) represents another critical choice. If a selected FF is inappropriate for a particular application, the result will tend to the best fitness, which may not perform best. In this sense, some common functions that evaluate the performance of the control system were tested. The options are Integral Absolute Error (IAE), Integral Square Error (ISE), and Integral Time Square Error (ITSE).

The test system is a parameterized second-order system:

\[ P(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \]  

(7)

When subjected to a step input, such a system presents a near-best transient when \( \zeta \approx 0.7 \). Then, it is reasonable to select a fitness that raises the higher value around \( \zeta = 0.7 \). Figure 4 shows the plots of fitness against \( \zeta \).

The Figure 4 evidence IAE has the peak close to 0.7. It also presents a good balance between settling time and overshoot. ITSE is also a good choice but puts a heavier weight on settling time, allowing for higher overshoot. If this is not the case, IAE is the best candidate for FF. In order to normalize the value of the FF, the following definition is adopted:

\[ \text{Fit} = \frac{1}{1 + \text{IAE}} \]  

(8)

where Fit approaches one as IAE tends to zero.
Figure 4. Fitness behavior of IAE, ISE, and ITSE as function of $\zeta$.

3. Test Plant Model

The application system comprises an axis with a rotating beam linked to a DC motor [31–35]. The control system must be tuned to perform well for load variation defined by different beams, which influences the total moment of inertia of the axis.

With the beams attached, the system is influenced by the drag force, as stated in Equation (9), which represents an opposite torque to the rotor movement [45].

$$ F_d = \frac{1}{2} C_d \rho A v^2 $$

where $\rho$ is the air density, $A$ is the area of the moving object facing the air, $C_d$ is the drag coefficient, and $v$ is the linear speed.

The state-space equations of the system are given by Equations (10) and (11)

$$ L \frac{di_a}{dt} = - Ri_a - k_e \omega + 11v_{pwm} $$

$$ J \frac{d\omega}{dt} = k_t i_a - B \omega - \frac{(k_{ms} \omega)^2 k_{acd}}{10} $$

where $L$ is the stator’s inductance, $R$ is the stator’s resistance, $J$ is the rotor’s moment of inertia, $B$ is the rotor’s friction coefficient, $k_e$ is the current-to-torque conversion constant, $k_t$ is the speed-to-induced-voltage conversion constant, $k_{ms}$ is the linear-to-rotational factor, $k_{acd}$ is the constant equivalent to $C_d \rho A$, $i_a$ is the stator’s current (state 1), $\omega$ is the rotor’s rotational speed (state 2), and $v_{pwm}$ is the input voltage in form of PWM pulses.

From Equations (10) and (11), a Simulink model was built in order to do computational simulations, as presented in Figure 5. The plant parameters are summarized in Table 1.

| Parameter | Value |
|-----------|-------|
| $L_a$     | 0.0032 H |
| $R_a$     | 4.57 $\Omega$ |
| $B$       | 0.001405 |
| $J$       | depends on the beam |
| $k_s$     | 0.25 |
| $k_{acd}$ | 0.02508 |
| $k_t$     | 0.23309 |
| $k_e$     | 0.23309 |
It is highlighted in Table 1 that the moment of inertia $J$ is not fixed because, in this work, it depends on the beam attached to the motor that acts as a variable load. Thus, the load is composed of the moments of inertia of the axis joint and a range of different beams, as summarized in Table 2. This range is used for simulations and covers the beams available in the physical plant.

Table 2. Moments of inertia used in simulations.

| Parameter | Value [kg·m²] |
|-----------|--------------|
| $J_{\text{j}}$ | 0.00026 |
| $J_1$ | 0.00250 |
| $J_2$ | 0.00400 |
| $J_3$ | 0.00550 |
| $J_4$ | 0.00700 |
| $J_5$ | 0.00850 |

4. Methodology

The methodology for developing this work is divided into two parts: the first determines the procedures for choosing the simulation parameters, starting with the proposal of 6 different scenarios for the GA; and then the six scenarios for the PSO. The best scenario found was used for an experimental evaluation, described in the second part of the methodology.

Thus, a computational model using Matlab/Simulink was elaborated for running simulations. This model comprises the plant and the GAPID controller.

This model is used during the optimization process, where all the algorithms’ variations are executed upon this model, and the desired results are extracted at the end of each simulation.

For each algorithm, each individual (or particle) needs to run the starting transient of the model for each iteration. This means that the model was executed once for $n$ individuals, with $k$ iterations, considering $h$ load variations, for all of the 12 algorithms.

The novelty here is that the load variations represent an additional variable to the optimization problem and then allow for finding an optimal and robust GAPID control. This is important because a large class of plants has variations in their operating conditions that must be considered to achieve a robust stabilization.
The final fitness results, as well as the individual evolution, were collected, tabulated and analyzed in order to verify which algorithm is more appropriate for the problem of finding the optimal parameters of the GAPID.

Some of the best results were tested in the physical plant, the DC motor-driven rotation beam, and one of them, the best one, is described in Section 5.

It is important to remark that the quantity of different loads utilized in the simulations is larger than that used in the physical plant, which allows for a greater definition of the load variation range during the optimization process, which occurs exclusively in the simulation environment.

4.1. Methodology for Simulation Analysis

4.1.1. GA Variations

In this work, 6 variations of the GA are used, all of them based on the premises described as follows [38,41,46]:

- GA1: The first GA follows the classic version initially proposed, except considering the mutation, which always occurs at a low rate. The roulette wheel is used to select the individuals that will participate in the crossover. The same individual may be selected more than once. The one-point crossover occurs with 70% probability, and 5% of the genes considering the entire population are randomly chosen and pass through the mutation. In this case, it is addressed as a perturbation considering a Gaussian distribution;
- GA2: The second GA is almost identical, but the crossover probability is 100%. It means that all selected individuals will generate offspring;
- GA3: This version uses the binary tournament instead of a roulette wheel. However, if an agent loses the game, it returns to the previous population and can be selected again;
- GA4: This version differs from the last because the “death tournament” is used. It means that if an agent loses the tournament, it is suppressed from the population;
- GA5: This proposal is more different from the others. The order of the operations is changed so that the crossover happens first, then the mutation, and finally the selection. With this idea, note that an intermediate subpopulation is created with the double individuals (see Section 2.2.1). Ultimately, the selection procedure (roulette wheel) is responsible for choosing the remaining agents, leaving the population with the original size. In this approach, all parents are chosen once at random to perform crossover;
- GA6: The last GA is similar to the previous one, but a tournament replaces the roulette wheel as a selection procedure.

4.1.2. PSO Variations

Two main topologies are proposed: in the Global Topology, each particle has its social factor $c_2$ related to the global best ($g_{best}$), while in the Ring Topology, the particle has its social factor related to its nearest neighbors [41].

Six variations of the PSO are addressed in this study:

- PSO1: The original proposal from Kennedy and Eberhart [43]. It uses Equations (5) and (6) considering the absent of $\omega$, or $\omega = 1$. The global topology is addressed.
- PSO2: It also uses the global topology but includes the inertial weight as a constant value between 0 and 1 [47]. Large values for the inertia are effective for global searching, while small values are better for local searching.
- PSO3: Another global topology, with a variable inertial weight, decreasing linearly from an initial value to a final one during the iterations, according to Equation (12), where

$$\omega = \omega_{max} - n \times \frac{\omega_{max} - \omega_{min}}{n_{max}}$$

(12)
where $\omega_{max}$ is the initial weight, $\omega_{min}$ is the final weight, $n$ is the iteration index and $n_{max}$ is the total iterations. This strategy provides a more efficient way to achieve a faster convergence in the last iterations due to gradually limiting the movement of the particles [48].

- PSO4: The same as PSO1, but using the ring topology [41].
- PSO5: The same as PSO2 with the ring topology.
- PSO6: The same as PSO 3 with the ring topology.

The summary of the examined strategies is presented in Figure 6.

![Figure 6. Six PSO strategies.](image)

After the simulation scan tests of the scenarios addressed above, the best result will be selected to evaluate it experimentally.

4.2. Methodology for Experimental Analysis

To experimentally evaluate the proposed solution, an experimental plant based on a DC Motor with variable load was developed.

The application system comprises an axis with a rotating beam linked to a DC motor. The control system must be tuned to perform well in a broad range of operating conditions. Figure 7 depicts the simplified schematic of the system.

![Figure 7. Schematic of the control system.](image)

The experimental plant was built with a DC Motor powered by a controlled voltage source passing through the Pololu Dual VN75019 motor driver shield module. This module is a compact breakout board for ST’s high-power motor driver IC, a fully integrated H-bridge to control the speed of a single brushed DC motor [49]. The motor is connected to an encoder to measure its rotation angle and speed. Figure 8a presents the image of both the motor drive module and the signal acquisition board, and Figure 8b shows the picture of the motor with the encoder attached.
Figure 8. (a) Motor system with beam; (b) motor with encoder.

The beams are interchangeable to simulate different loads. Three test beams were used: light slim aluminum, carbon steel, and thick heavy aluminum, as shown in Figure 9. The moments of inertia for these beams are presented in Table 3.

Table 3. Moments of inertia for the three beams for experimental tests.

| Beam (Load)                  | Inertia Value (J) [kg·m²] |
|------------------------------|---------------------------|
| Light slim aluminum beam     | 0.0035                    |
| Carbon steel beam            | 0.0077                    |
| Heavy thick aluminum beam    | 0.0080                    |

Data from the encoder are sent to a PLC CompactRIO®, a real-time embedded industrial Data Acquisition and controller made by National Instruments for industrial control systems. This system consists of a controller with a microprocessor and an FPGA programmable by the user. In our case, the encoder (sensor) signal is connected as a digital input and the PWM signal to control the motor speed as an analog output [50]. Figure 10 illustrates the PLC used.
5. Results

5.1. Simulation Results

As stated in Section 2.2, six GAs and six PSOs were tested to study the behavior of each algorithm. Each algorithm was run 10 times with the average results taken to compose the figures and tables reported in the following sections. The objective is to observe the characteristics, performance, advantages, and drawbacks of each strategy. Simulations were carried out in Matlab/Simulink employing GA and PSO optimization algorithms. The first step is to compare GA and PSO strategies separately. This action is needed to reach the behavior of selecting different parameters within the same algorithm. In the second step, a general comparison between these two algorithms is made, particularly for the final result, remembering that due to their nature being so different, no fair comparison can be realized regarding the evolution of each algorithm.

5.2. Comparison of GA Strategies

As stated in Section 2.2, six GAs were tested to study the behavior of each algorithm. The objective is to observe the characteristics, performance, advantages, and drawbacks. The parameters of each GA employed in the tests are summarized in Table 4.

| Parameter          | GA1    | GA2    | GA3    | GA4    | GA5    | GA6    |
|--------------------|--------|--------|--------|--------|--------|--------|
| Sequence           | S+C+M  | S+C+M  | S+C+M  | S+C+M  | C+M+S  | C+M+S  |
| Selection          | RW     | RW     | BT     | DT     | RW     | BT     |
| Crossover probability | 70%   | 100%   | 70%    | 70%    | 100%   | 100%   |
| Mutation           |        |        |        |        | 5%     |        |
| Population         |        |        |        |        | 30     |        |
| Repetitions        |        |        |        |        | 10     |        |

1 S = selection, C = crossover, M = mutation. 2 RW = roulette wheel, BT = binary tournament, DT = death tournament.

Figure 11 presents the evolution of each GA regarding the best, the worst, and the mean of all individuals in the population.
Figure 11. Evolution of GA strategies.

To better analyze the overall population distribution at the end of iterations, a boxplot of each GA variant was done, shown in Figure 12.

The final results with the best fitness values are summarized in Table 5.
Table 5. The best fitness of each GA.

| GA model | Fitness  | Rank |
|----------|----------|------|
| GA1      | 0.993574 | 5    |
| GA2      | 0.993661 | 4    |
| GA3      | 0.993898 | 2    |
| GA4      | 0.993680 | 3    |
| GA5      | 0.993432 | 6    |
| GA6      | 0.993999 | 1    |

From Figure 12 and Table 5, some conclusions can be taken: GA3, GA4, and GA6 presented less-dispersed populations in higher fitness ranges, with a smaller dispersion in GA4. This last finding in GA4 may pose a problem because some dispersion is often expected in GAs. The main similarity of these strategies is the use of binary tournament selection. On the other hand, GA1 and GA5, which use the roulette wheel, showed worse results with higher dispersion.

It seems clear that the use of roulette wheel should be avoided in this problem. The literature is abundant in presenting the drawbacks of applying such a selection procedure because it presents a high selective pressure. It means that the individuals with the highest fitness present a high probability of being selected at each iteration. However, this procedure may prematurely guide the search for local minimum points, often far from the global optimum. In this sense, the tournament is a better candidate.

In addition, the use of the death tournament did not perform better than the traditional binary tournament due to the risk of losing potentially well-positioned agents. As seen in Figure 12, despite the dispersion being small in GA4, the top fitness is not the highest.

Regarding creating a sub-population, if the strategy is used with a binary tournament, the results revealed that the algorithm could reach better values to the parameters of the GAPID due to the existence of more individuals simultaneously. This procedure may increase the probability of finding good positions in the search space when applying the genetic operators.

Despite presenting the highest dispersion, GA2 achieved a result comparable to GA3, GA4, and GA6 in terms of the mean and the best individuals. Finally, it is important to mention that the results lie within the topmost 1% of the normalized fitness values. This means that all GAs perform quite well, with slightly different results.

5.3. Comparison of PSO Strategies

In addition to the analysis of GA algorithms, six PSO variations were evaluated. The parameters were determined by previous tests and are detailed in Table 6.

Table 6. PSO parameters.

| Parameter       | PSO1 | PSO2 | PSO3 | PSO4 | PSO5 | PSO6 |
|-----------------|------|------|------|------|------|------|
| Topology        | Global | Ring |
| $\omega$        | 1    | 0.5  | 0.9 → 0.4 | 1    | 0.5  | 0.9 → 0.4 |
| Search space    | 100  |
| Population      | 30   |
| Dimensions      | 6    |
| Maximum iterations | 50   |
| $c_1$ and $c_2$ | 2.05 |
| Repetitions     | 10   |

As mentioned in Section 4.1.2, PSO 1, 2, and 3 use the global topology while PSO 4, 5, and 6 use the ring topology; variants 1 and 4 use no inertial weight $\omega$ while 2 and 5 use fixed $\omega$ and 3 and 6 use the linear decreasing weight approach [41].
Table 7 presents the best fitness obtained after 10 simulations for each PSO strategy. It can be noted that the first-ranked strategies employ the global topology, which reinforces the conclusion that this is the best topology. Indeed, for this specific application, the full communication between the particles and the best-positioned one is an advantage, leading to better configurations. It seems clear that the ring topology now allowed the search for the maximum fitness points.

Table 7. The best fitness of each PSO

| PSO Model | Best Fitness | Rank |
|-----------|--------------|------|
| PSO1      | 0.992541     | 3    |
| PSO2      | 0.992565     | 2    |
| PSO3      | 0.992588     | 1    |
| PSO4      | 0.992280     | 4    |
| PSO5      | 0.990775     | 5    |
| PSO6      | 0.990427     | 6    |

Considering the best combination of strategies used to set $\omega$, linear decay proved to be the best choice, which is an important finding since the modification in the value of $\omega$ along the iterations adjusts the model to perform a kind of global search strongly at the beginning of the process (exploration) and to perform a local search refining the solutions along the process (exploitation).

Interestingly, while the global topology led the swarm to small dispersion using all chosen values of $\omega$, its application in the ring topology degraded the performance and increased the dispersion. This is an unexpected behavior since the algorithm tends to present a problematic convergence. This finding reinforces the need to choose the best configuration, considering the specificities of the topology selected.

The Friedman test was applied to the results and resulted in a $p$-value of 0.0027, which means that an optimizer change leads to different results.

Considering the parameters presented in Tables 5 and 7, simulations were carried out, and all the results were summarized in a boxplot, presented in Figure 13.

Figure 13. Boxplot of the six PSO strategies.

This fact evidences the dispersion of results and allows for some statistical analysis that help to assess the differences among the PSO versions.

Due to the nature of the chosen normalized fitness function in Equation (8), it is important to remark that small values of the cost function IAE result in fitness values close
to 1. So, even if the values presented in the graph seem comparable, the performance differences are considerable.

At a first glance, it is clear that the global topology is better for this problem. The performances in these approaches are better than with their ring topology counterparts. The dispersion in these variants is also narrower, which means these algorithms achieve better solutions due to their exploration capability [41].

Comparing the best three proposals demonstrates that using an inertial weight with linear decrease seems to be an advantage.

Another observation is that the topology change results in special performances related to updating velocities. In the most straightforward velocity update strategy, the topology of the ring generates the best results.

Finally, PSOs 5 and 6 are the worst regarding fitness and dispersion.

Figure 14 shows the evolution of each PSO strategy for each iteration. It gives an idea of how the individuals behave during the search. The graphics show the Global Best, the best in each iteration (Iteration Best), the average best of the iteration (Average), and the worse of the iteration (Iteration Worst).

Figure 14. Evolution of PSO strategies.

In most cases, 30 iterations are enough for convergence. By the evolution of the average best, PSO2 and PSO3 achieved faster Average and Global Best convergences, indicating fewer iterations would be necessary for the optimization.

The resulting best parameters found, in this case by PSO3, are summarized in Table 8. These parameters were selected and employed to obtain the waveforms shown in Figure 15 and the experimental tests presented in the next section.
It is known that PID has limited robust responses for parameter variations, primarily due to its linear nature, which those nonlinear controllers can overcome. Then, a load sweep was performed to analyze each behavior of the controller, PID, and GAPID, remembering that the GAPID parameters are linked to the PID gains.

Table 8. Best GAPID parameters found (by PSO3).

| Gain type  | Gaussian Parameters |
|------------|---------------------|
| Proportional | \( k_p = 0.2 \) | \( q_p = 0.9 \) |
| Integral   | \( k_i = 0.0637 \) | \( q_i = 0.9 \) |
| Derivative | \( k_d = 14 \)      | \( q_d = 0.001 \) |

In Figure 15, the output transients of the PID and GAPID controlled system subjected to variation of the moment of inertia in the range from 0.002 to 0.015 are presented. As expected, the load variation causes a significant change in behavior in the linear PID. In the GAPID, on the other hand, this behavior change is minimal, providing good robustness to load variation. In this case, there was also a considerable performance improvement.

It is important to remark that, in our case, the fitness function is based on IAE, and load variation is considered, showing the robustness behavior becomes the primary objective of the optimization.

![Figure 15. Transient responses of the system with PID and GAPID controllers subjected to variation of the moment of inertia \( J \) from 0.002 to 0.015.](image)

5.4. Experimental Results

For the experimental analysis, the three beams presented in Section 4.2 were used as loads to the presented physical plant. In it, the motor carries different beams which represent different loads to test the performance and robustness of the controller.

The PID and the GAPID-tuned controllers were tested to a set point of angular speed of 5 rad/s, with the startup waveforms shown in Figures 16 and 17.

For the PID controller, the gain values used were 0.2578 for the proportional \( (k_p) \), and 0.0625 for the integral \( (k_i) \), and the output response for each of the three beams is presented in Figure 16.

In it, one can see that the output speed kept an overshoot under 5% and a settling time under 2 s with the light aluminum beam it was projected for. In contrast, the two other
beams could not maintain a low overshoot and took longer to stabilize. The system was not unstable, and the PID was still able to control, but not with the same performance.

For the parameters of GAPID, the best solution (PSO3) parameters were taken, representing the set of parameters shown in Table 8.

In Figure 17, although the slower settling time is close to 4 s, the overshoot was under 5% for the three beams. It is noticeable that the output response behaves similarly even with the load mass variation in comparison to the performance of the PID.

![Figure 16. DC motor angular speed with PID control for different loads, to set point of 5 rad/s.](image1)

![Figure 17. DC motor angular speed with GAPID control for different loads, to set point of 5 rad/s.](image2)

It can be observed that the startup transients shown in Figure 16 demonstrate that only the designed PID for one specific load performs well, while the others exhibit a lack of robustness. Again, the robustness advantage of GAPID is evident, as shown in Figure 17, where the controller performs well for all loads.

6. Conclusions

This work addressed a comparative study about two bio-inspired algorithms, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), with six strategy variations each. These algorithms were used to find the optimal parameters of the Gaussian Adaptive PID control (GAPID) controller.
The test plant is a controlled rotating beam driven by a DC motor. The mathematical model was derived, enabling the building of a computational model used in simulations, and an experimentation plant was used to validate the results.

The GAPID, used to control this plant, represents an NP-Hard and multi-modal optimization problem with several distinct suboptimal solutions, then metaheuristics are used to find optimal solutions to the controller parameters.

In the PSO, the strategies were divided by topology (global or ring topology) and inertial weight factor (absent, fixed, or linearly decreasing).

The quality of the solution was evaluated by a fitness function that employs the Integral Absolute Error (IAE) value. This index covers the settling time and the overshoot of the output waveform.

The computational results were presented in boxplots which allow the analysis of the quality and dispersion of the solutions. It demonstrated that the global topology performs better than the ring topology, with higher fitness values and lower distribution.

A step response test shows how each GAPID optimization performs and how much better they are concerning PID. It is an essential conclusion since the existence of adaptive, robust, and reliable control strategies are very important in real applications.

In future works, it is planned to study a multi-objective optimization approach, which may also help us to better select robust solutions among the near best results. Moreover, other metaheuristic strategies can be tested to determine if any of them are more suitable for this problem, such as Whale Optimization Algorithm (WOA), Artificial Bee Colony (ABC), Differential Evolution (DE), Firefly Algorithm (FA), Bat Algorithm (BA), among others, should be tested in future works. Finally, other plants are also planned to be used as test plants, including higher-order and nonlinear ones.

**Author Contributions:** Conceptualization: H.V.S. and M.S.K.; methodology: F.C.J. and P.E.S.M.; validation: F.C.C., P.E.S.M. and M.S.K.; formal analysis: F.C.C. and M.d.S.K.; investigation: F.G.B. and M.S.K.; resources: F.C.C., F.C.J. and M.S.K.; writing—original draft preparation: M.G., P.E.S.M. and F.G.B.; writing—review and editing: S.L.S.J., M.S.K. and H.V.S.; visualization: S.L.S.J.; supervision: H.V.S.; funding acquisition: H.V.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Coordination for the Improvement of Higher Education Personnel (CAPES)—Financing Code 001, Brazilian National Council for Scientific and Technological Development (CNPq), processes number 40558/2018-5, 315298/2020-0, and Araucaria Foundation, process number 51497.

**Acknowledgments:** The authors thank the Federal University of Technology–Parana, and Brazilian agencies: Coordination for the Improvement of Higher Education Personnel (CAPES), Brazilian National Council for Scientific and Technological Development (CNPq) and Araucaria Foundation.

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Puchta, E.D.P.; Siqueira, H.V.; Kaster, M.S. Optimization Tools Based on Metaheuristics for Performance Enhancement in a Gaussian Adaptive PID Controller. *IEEE Trans. Cybern.* **2019**, *50*, 1185–1194. [CrossRef] [PubMed]
2. Atherton, D.P.; Majhi, S. Limitations of PID controllers. In Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), San Diego, CA, USA, 2–4 June 1999; Volume 6, pp. 3843–3847. [CrossRef]
3. Sung, S.W.; Lee, I.B. Limitations and countermeasures of PID controllers. *Ind. Eng. Chem. Res.* **1996**, *35*, 2596–2610. [CrossRef]
4. Palanikumar, A.; Saikumar, N.; HosseinNia, S.H. No more differentiator in PID: Development of nonlinear lead for precision mechatronics. In Proceedings of the 2018 European Control Conference (ECC), Limassol, Cyprus, 12–15 June 2018; pp. 991–996. [CrossRef]
5. Hsia, T. Adaptive control of robot manipulators—A review. In Proceedings of the 1986 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, 7–10 April 1986; Volume 3, pp. 183–189. [CrossRef]
6. Astrom, K.J.; Wittenmark, B. *Adaptive Control*, 2nd ed.; Dover Publications: New York, NY, USA, 2008.
7. Shaocheng, T.; Jiantao, T.; Tao, W. Fuzzy adaptive control of multivariable nonlinear systems. *Fuzzy Sets Syst.* **2000**, *111*, 153–167. [CrossRef]
8. Golea, N.; Golea, A.; Benmahammed, K. Stable indirect fuzzy adaptive control. *Fuzzy Sets Syst.* **2003**, *137*, 353–366. [CrossRef]
9. Hunt, K.J. *Neural Adaptive Control Technology;* World Scientific: Singapore, 1996; Volume 15.
10. Dracopoulos, D.C. *Evolutionary Learning Algorithms for Neural Adaptive Control;* Springer: Berlin/Heidelberg, Germany, 2013.
11. Xu, H.; Mirmirani, M.; Ioannou, P. Robust neural adaptive control of a hypersonic aircraft. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit,* Austin, TX, USA, 11–14 August 2003; p. 5641.
12. Wang, Z.; Zhang, Y.; Fang, H. Neural adaptive control for a class of nonlinear systems with unknown deadzone. *Neural Comput. Appl.* 2008, 17, 339–345. [CrossRef]
13. Jinli, L. Adaptive Fuzzy PID Control for Boiler Deaerator. In *Proceedings of the 2012 International Conference on Industrial Control and Electronics Engineering,* Xi’an, China, 23–25 August 2012; pp. 575–578. [CrossRef]
14. Korkmaz, M.; Aydogdu, O.; Dogan, H. Design and performance comparison of variable parameter nonlinear PID controller and genetic algorithm based PID controller. In *Proceedings of the 2012 International Symposium on Innovations in Intelligent Systems and Applications,* Trabzon, Turkey, 2–4 July 2012; pp. 1–5. [CrossRef]
15. Qi, W.M.; Cai, W.Y.; Ji, Q.L.; Cheng, Y.C. A Design of Nonlinear Adaptive PID Controller Based on Genetic Algorithm. In *Proceedings of the 2006 Chinese Control Conference,* Harbin, China, 16–18 August 2006; pp. 175–178. [CrossRef]
16. Lucas, R.; Oliveira, R.M.; Nascimento, C.B.; Kaster, M.S. Performance analysis of an adaptive Gaussian nonlinear PID control applied to a step-down CC-CC converter. In *Proceedings of the 2015 IEEE 24th International Symposium on Industrial Electronics (ISIE),* Rio de Janeiro, Brazil, 3–5 June 2015; pp. 1022–1026. [CrossRef]
17. Puchta, E.D.P.; Lucas, R.; Ferreira, F.R.V.; Siqueira, H.V.; Kaster, M.S. Gaussian adaptive PID control optimized via genetic algorithm applied to a step-down DC-DC converter. In *Proceedings of the 2016 12th IEEE International Conference on Industry Applications (INDUSCON),* Curitiba, Brazil, 20–23 November 2016; pp. 1–6.
18. Guerreiro, M.T.; Guerreiro, E.M.A.; Barchi, T.M.; Biluca, J.; Alves, T.A.; de Souza Tadano, Y.; Trojan, F.; Siqueira, H.V. Anomaly Detection in Automotive Industry Using Clustering Methods—A Case Study. *Appl. Sci.* 2021, 11, 9868. [CrossRef]
19. Tao, L.; Yang, X.; Zhou, Y.; Yang, L. A novel transformers fault diagnosis method based on probabilistic neural network and bio-inspired optimizer. *Sensors* 2021, 21, 3623. [CrossRef]
20. Santos, P.; Macedo, M.; Figueiredo, E.; Santana, C.J.; Soares, F.; Siqueira, H.; Maciel, A.; Gokhale, A.; Bastos-Filho, C.J. Application of PSO-based clustering algorithms on educational databases. In *Proceedings of the 2017 IEEE Latin America Conference on Computational Intelligence (LA-CCI),* Arequipa, Peru, 8–10 November 2017; pp. 1–6. [CrossRef]
21. Zivkovic, M.; Strean, C.; Chhabra, A.; Budimirovic, N.; Petrovic, A.; Bacanin, N. Novel improved salp swarm algorithm: An application for feature selection. *Sensors* 2022, 22, 1711. [CrossRef]
22. Pitchaimanickam, B.; Murugaboopathi, G. A hybrid firefly algorithm with particle swarm optimization for energy efficient optimal cluster head selection in wireless sensor networks. *Neural Comput. Appl.* 2020, 32, 7709–7723. [CrossRef]
23. Devan, P.A.M.; Hussin, F.A.; Ibrahim, R.B.; Singi, K.; Nagarajapandian, M.; Assaad, M. An arithmetic-trigonometric optimization algorithm with application for control of real-time pressure process plant. *Sensors* 2022, 22, 617. [CrossRef] [PubMed]
24. Lin, J.; Zheng, R.; Zhang, Y.; Feng, J.; Li, W.; Luo, K. CFHBA-PID Algorithm: Dual-Loop PID Balancing Robot Attitude Control Algorithm Based on Complementary Factor and Honey Badger Algorithm. *Sensors* 2022, 22, 4492. [CrossRef] [PubMed]
25. Huba, M.; Vrancic, D. Performance Portrait Method: An Intelligent PID Controller Design Based on a Database of Relevant Systems Behaviors. *Sensors* 2022, 22, 3753. [CrossRef] [PubMed]
26. Kang, C.; Park, B.; Choi, J. Scheduling PID Attitude and Position Control Frequencies for Time-Optimal Quadrotor Waypoint Tracking under Unknown External Disturbances. *Sensors* 2021, 22, 150. [CrossRef] [PubMed]
27. Belotti, J.T.; Castanho, D.S.; Araujo, L.N.; da Silva, L.V.; Alves, T.A.; Tadano, Y.S.; Stevan S.L., Jr.; Corrêa, F.C.; Siqueira, H.V. Air pollution epidemiology: A simplified Generalized Linear Model approach optimized by bio-inspired metaheuristics. *Environ. Res.* 2020, 191, 110106. [CrossRef]
28. Puchta, E.D.P.; Bassetto, P.; Biuk, L.H.; Itaborahy Filho, M.A.; Converti, A.; Kaster, M.S.; Siqueira, H.V. Swarm-inspired algorithms to optimize a nonlinear gaussian adaptive PID controller. *Energies* 2021, 14, 3385. [CrossRef]
29. Puchta, E.P.; Lucas, R.; Siqueira, H.V.; Kaster, M.S. Application of a Nonlinear Gaussian Adaptive PID with PSO Optimization to a Buck Converter. In *Proceedings of the Congresso Brasileiro de Automatica,* Sociedade Brasileira de Automatica, Vitoria, Brazil, 3–7 October 2016.
30. Bassetto, P.; Puchta, E.D.P.; Biuk, L.H.; Itaborahy Filho, M.A.; Kaster, M.S.; Siqueira, H.V. Metaheuristic-based optimization applied to GAPID controller. In *Proceedings of the 2021 14th IEEE International Conference on Industry Applications (INDUSCON),* Sao Paulo, Brazil, 15–18 August 2021; pp. 820–827. [CrossRef]
31. Janzen, F.C.; Tusset, A.M.; Balthazar, J.M. Positioning Control of a Flexible Slewing Structure by Applying Sliding Mode Control. In *Proceedings of the 2021 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* Charlotte, NC, USA, 21–24 August 2016. [CrossRef]
32. Hein, R.; Orlikowski, C. Simplified dynamic model of rotating beam. *Diagnostika* 2013, 14, 43–48.
33. Nandi, A.; Neogy, S.; Iretire, H. Vibration Control of a Structure and a Rotor Using One-sided Magnetic Actuator and a Digital Proportional-derivative Control. *J. Vib. Control* 2009, 15, 163–181. [CrossRef]
34. Sarkar, S.; Nandi, A.; Neogy, S. Control of Simulink Based Jeffcott Rotor Model. In *Proceedings of the 6th International Conference on Vibration Engineering and Technology on Machinery (VETOMAC 2010),* New Delhi, India, 13–15 December 2010.
35. Mattja, T.; Lazarz, B. Modeling the coupled flexural and torsional vibrations in rotating machines in transient states. *J. Vibroeng.* 2014, 16, 1911–1924.
36. Ippoliti, E. Heuristic Reasoning; Springer: Berlin/Heidelberg, Germany, 2014; Volume 16.
37. Bianchi, L.; Dorigo, M.; Gambardella, L.M.; Gutjahr, W.J. A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. 2009, 8, 239–287. [CrossRef]
38. Holland, J.H. Adaptation in Natural and Artificial Systems; MIT Press: Cambridge, MA, USA, 1992.
39. Figueiredo, E.; Macedo, M.; Siqueira, H.V.; Santana, C.J., Jr.; Gokhale, A.; Bastos-Filho, C.J. Swarm intelligence for clustering—A systematic review with new perspectives on data mining. Eng. Appl. Artif. Intell. 2019, 82, 313–329. [CrossRef]
40. Michalewicz, Z. Genetic Algorithms + Data Structures = Evolution Programs; Springer: Berlin/Heidelberg, Germany, 1996.
41. Castro, L.N. Fundamentals of Natural Computing: Basic Concepts, Algorithms and Applications; Chapman & Hall/CRC: Boca Raton, FL, USA, 2006.
42. Goldberg, D. The Design of Innovation (Genetic Algorithms and Evolutionary Computation); Springer: London, UK, 2002.
43. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks IV, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
44. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. 2002, 6, 58–73. [CrossRef]
45. Janzen, F.C. Positioning and Vibration Control of a Flexible Structure in Slewing Motion by Applying Shape Memory Alloys. Ph.D. Thesis, Universidade Estadual Paulista, São Paulo, Brazil, 2016.
46. Siqueira, H.; Belotti, J.T.; Boccato, L.; Luna, I.; Attux, R.; Lyra, C. Recursive linear models optimized by bioinspired metaheuristics to streamflow time series prediction. Int. Trans. Oper. Res. 2020. [CrossRef]
47. Eberhart, R.C.; Shi, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), La Jolla, CA, USA, 16–19 July 2000; Volume 1, pp. 84–88. [CrossRef]
48. Khan, T.A.; Taj, T.A.; Asif, M.K.; Ijaz, I. Modeling of a standard Particle Swarm Optimization algorithm in MATLAB by different benchmarks. In Proceedings of the Second International Conference on the Innovative Computing Technology (INTECH 2012), Casablanca, Morocco, 18–20 September 2012; pp. 271–274. [CrossRef]
49. Pololu. Pololu Dual VNH5019 Motor Driver Shield for Arduino. 2019. Available online: https://www.pololu.com/product/2507 (accessed on 17 May 2019).
50. Instruments, N.N. CompactRIO Systems. 2019. Available online: http://www.ni.com/pt-br/shop/compactrio.html (accessed on 22 May 2019).