Received June 13, 2020, accepted June 23, 2020, date of publication June 26, 2020, date of current version July 8, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.3005318*

# Recent Meta-Heuristics Improved by Self-Adaptation Applied to Nonlinear Model-Based Predictive Control

**EDUARDO DE MENDONÇA MESQUITA[1], RENATO CORAL SAMPAIO[2,3], (Member, IEEE), HELON VICENTE HULTMANN AYALA[4], AND CARLOS HUMBERTO LLANOS[3], (Member, IEEE)**

[1]School of Electric and Computer Engineering, Federal University of Goiás, Goiânia 74605-010, Brazil
[2]Faculty of Gama, University of Brasília, Brasília 72444-240, Brazil
[3]Faculty of Technology, University of Brasília, Brasília 70.910-900, Brazil
[4]Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro 22451-900, Brazil

Corresponding author: Eduardo de Mendonça Mesquita (eduardo.dmendonca@gmail.com)

This work was supported in part by the National Council for Scientific and Technologic Development (CNPq), and in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior(CAPES).

**ABSTRACT** Nonlinear Model-based Predictive Control (NMPC) is a relevant research area having applications in the industrial sector. Traditionally, in this technique, gradient descent algorithms have been used to solve the related optimization problem. More recently, bio-inspired meta-heuristics have also been applied to this problem. However, only a few works have been devoted to testing solvers that use parameter control with self-adaptive traits, which allows mitigating the problem of offline parameter tuning in bio-inspired approaches. In this paper, we propose the novel Adaptive Modified Grey Wolf Optimization (AMGWO) and the Adaptive Moth-Flame Optimization (AMFO), for solving Nonlinear Model-based Predictive Control (NMPC) problems. To achieve this, a mechanism for individual leaders weighting and a crossover operator are introduced in AMGWO, and a simple self-adaptive parameter technique is applied in both meta-heuristics. The improved solvers are tested to perform the swing-up of a single inverted pendulum and attitude control of a satellite, which are nonlinear problems relevant for assessing control performance. Nonparametric statistical tests are applied to compare the improved meta-heuristics optimization outcomes with other five meta-heuristics, which shows that the self-adaptive parameter technique can significantly improve the performance when applied as an NMPC solver, as the AMFO and AMGWO statistically outperform or performs as well as all algorithms compared in both the pendulum and satellite control, respectively. This is important as improving the optimizer efficiency will lead to more accurate control and enable rapid hardware implementation.

**INDEX TERMS** Adaptive algorithms, computation intelligence, nonlinear dynamical systems, predictive control.

## I. INTRODUCTION

The Model-based Predictive Control (MPC) is a well-known control technique widely used in industrial systems. The MPC performs the control by calculating the optimal actions based on the predicted future behavior of a given plant through an online optimization process [1]. Gradient descent algorithms have been used for the purpose of solving these optimization problems [2]–[4] but it has been shown to be limited in computational capacity for nonlinear and non-convex problems [5]. Regarding this, Bio-inspired Meta-heuristics (BMs) proved to be an alternative for dealing with these complex problems [6], for example in advanced control [7], [8]. BMs' ability to explore and exploit in linear, non-linear, and multi-objective problems makes it useful in various classes of MPC, including linear MPC, nonlinear MPCs (NMPCs), tracking-based MPCs, hybrid MPCs, among others. This fact presents another advantage of BMs over Traditional Algorithms (TAs) since the latter has the primary ability to solve convex optimization problems [5]. One of the challenges in designing and implementing NMPC in a real-world context is to seek algorithms capable of obtaining

The associate editor coordinating the review of this manuscript and approving it for publication was Bohui Wang.

the optimal control actions for a given system since different models require different solutions.

BMs are a subset of *soft-computing* methods that use biological phenomena present in natural systems as metaphors to their heuristic strategies to search for solutions of optimization problems [9]. What makes meta-heuristics advantageous concerning the exact optimization methods are their characteristics of simplicity and flexibility. In terms of simplicity, they are based on simple mathematical models, generally inspired in natural phenomena, looking for balancing two basic processes such as *exploring* a broad range of possible solutions by different search agents, and *exploiting* the information gathered by them to avoid potential local optima. This balance is essential to prevent phenomena such as premature convergence and stagnation of the process in local minima/maximums. In terms of flexibility, meta-heuristics can be easily adapted to different types of problems, without having to take prior knowledge of the forms of the cost function, avoiding calculating derivatives, such as with the exact optimization methods. These characteristics make them suitable also in dynamic situations where the goal or constraints are changing over time, either for exogenous or self-induced causes, and the parameter adjustments and fitness measurements are disturbed. The same is true when the search space is multidimensional, multimodal, or fractal and cannot be handled by traditional methods, especially those that use a global prediction from local surface analysis [10].

In general, BMs are categorized into two groups based on their metaphors: (a) *evolutionary* and (b) *swarm intelligence* algorithms. Evolutionary techniques are inspired by Darwinian evolution theory that describes the method in which a species population changes their individuals genetic material through generations as an adaptation to the environment, the best known being the Genetic Algorithm (GA) [11], Differential Evolution (DE) [12], and their variations. Otherwise, *swarm intelligence* ones use as inspiration for their metaphors the social behavior displayed by some species of animals. Techniques, in this category, generally encode a possible solution as the position of an individual in the group (swarm) and use the evaluation of the objective function to force the movement of individuals towards improved solutions. Examples of this proposal are Particle Swarm Optimization (PSO) [13], Salp Swarm Algorithm (SSA) [14], Grey Wolf Optimization (GWO) [15], Elephant Herd Optimization (EHO) [16], Dragonfly Algorithm (DA) [17], and Moth-Flame Optimization (MFO) [18].

The interest of the scientific community in meta-heuristic methods, including the intense generation of new methods in the past decade, uses as theoretical base the *No Free Lunch* (NFL) Theorem. The NFL theorem [19] shows that all meta-heuristic algorithms have the same performance on average when applied to all possible problems. In other words, if a meta-heuristic $BM_1$ achieves better results than another $BM_2$ when applied to a set of problems, $BM_1$ performs worse when applied on another set of problems, where $BM_2$ performs better. For this reason, to decide what is the best bio-inspired meta-heuristic to optimize a complicated family of optimization problems, it is essential to use and compare a significant number of meta-heuristics on a range of combinations of problems.

BM algorithms' performance is still heavily dependant on the setting of its design parameters. Tuning them is far from straightforward and can be manually impractical. There are two possible modes for parameters value defining strategies in meta-heuristics: (a) *parameter tuning*, and (b) *parameter control* [20]. Parameter tuning is a strategy to search for near-optimal values for the design parameters of a heuristic algorithm. This search is generally an empirical trial-and-error method (usually performed offline), where the whole optimization process is repeated multiple times to find parameters that bear good results when compared with others. Otherwise, parameter control is a self-adaptation alternative that adds techniques that are capable of dynamically changing parameter values during a meta-heuristic execution. This alternative mitigates the dependence on a proper time-consuming initial parameter tuning while improving the algorithm performance.

Motivated by this consideration, different adaptive and self-adaptive mechanisms [21]–[26] have arisen in order to dynamically adapt the parameters without the user having prior knowledge of the relationship between parameter setting and the characteristics of optimization problems. Therefore, the parameter adaptation, if well designed, is capable of improving an algorithm's convergence performance. The self-adaptation mechanisms are mainly applied in classic optimization problems or benchmark functions, whereas optimal control policies such as NMPC applications still lack in the literature.

This paper focus on two aforementioned BM's types: the GWO [15], which is based on social rank and hunting behavior of wolves and the MFO [18], based on lighting attraction of moths. Since GWO and MFO are widely applied in optimization tasks, several improvement methods were proposed in these meta-heuristics to obtain better performance. For instance, to GWO were applied a cauchy operator [27], [28], chaotic mapping [29]–[31], fuzzy logic [32] and refraction learning [33]. Improved MFO versions were utilized in parameter tuning problems [34]–[36], feature selection [37], photo-voltaic models [38], standard benchmark functions [39], [40], power flow problem [41], among others. Self-adaptation was applied in GWO to solve a transmit antenna selection problem [42] and a 2-dimensional logistic chaotic mapping [43]. A modified MFO by introducing a self-adaptive inertia weight was used to solve a multilevel thresholding segmentation for a color image problem [44]. In addition, an adaptive mutation and multi-parent crossover in binary GWO is found in [45].

The use of BMs as MPC solvers has been extensively explored and has been shown well-suitable to obtain control variables online, with accurate control precision and in great robustness. Many MPC approaches have been reported in the literature using BMs. DE meta-heuristic has been used as

MPC solver in a pressure control system [46] and position control of a robotic arm model [47]. Power systems [48], engine idle speed [49] problems and a formation control of multiple unmanned aerial vehicles [50] was solved by a PSO-based MPC scheme. A hybrid GWO-based MPC was implemented to control of aircraft engines [51], and an improved GWO with an individual memory fitness based on PSO was applied in an MPC strategy for trajectory optimization of real-time multi-UAV target [52]. Other types of soft computing methods have been reported to solve NMPC. To cite a few, a radial basis function neural network was used as an MPC solver to reduce structural fatigue loads in large wind turbines [53], a simplified dual neural network-based MPC was implemented on a digital signal processor device and applied to an air-separation unit system [54] and a general projection neural networks works as NMPC solver applied in a multi-robot formation control [55]. However, the applicability of self-adaptation BMs in NMPC approaches is little addressed in the literature [56]–[58], and the proposal/evaluation of simple self-adaptation methods are not encountered in this context.

Therefore, the main contributions of this work are the following:

(i) the proposal of a self-adaptive parameter technique to both GWO and MFO meta-heuristics. To achieve this, we have analyzed and adapted the application of the parameter self-adaptation proposed in [22], to improve the performance of two original meta-heuristics;
(ii) the inclusion of a new mechanism for weighting individual leaders at each interaction in the GWO in order to improve the guidance method of best individuals and to obtain a more efficient exploitation phase;
(iii) the introduction of a new crossover operator in the GWO to improve the meta-heuristic convergence speed-up;
(iv) the implementation of the proposed meta-heuristics as NMPC solvers in two nonlinear control systems where their performances are compared with other five meta-heuristics, using non-parametric statistical tests, such as in [59]–[62].

The new meta-heuristics have been called Adaptive Modified Grey Wolf Optimizer (AMGWO) and Adaptive Moth Flame Optimization (AMFO). The experiment results show evidence that, for the set of problems used, our proposed self-adaptive bio-inspired meta-heuristics statistically outperforms or performs as well as all algorithms compared, without the necessity of an off-line parameter tuning, when compared with the ones that do no contain adaptive mechanisms, or even with previous seminal auto-adaptive proposals (e.g., JADE [22] and LSHADE [26]). These results are promising to the application of meta-heuristics to NMPC.

The paper is organized as follows. Section II presents the meta-heuristic optimization methods as they were first proposed. Then, self-adaptation versions are detailed in Section III. The NMPC algorithm and the problem definitions are stated in Section IV. Next, the simulation results and their corresponding analysis are given in Sections V and VI, respectively, to show the effectiveness of the newly proposed meta-heuristics. Finally, conclusions are summarized in Section VII.

## II. META-HEURISTIC OPTIMIZATION TECHNIQUES TO BE IMPROVED

In the present study, we have analyzed two recent meta-heuristics presented in the literature, which are: (1) Grey Wolf Optimizer - GWO [15] inspired by the behavior of the pack of grey wolves (Canis Lupus) and (2) Moth-Flame Optimization - MFO [18] based on the moonlight-guided locomotion method.

In order to accredit our study, three other meta-heuristics were analysed: (1) PSO with the *inertia factor* [63], (2) JADE [22] and (3) LSHADE [26], the last ones representing two self-adaptation versions of the classical meta-heuristic DE [12].

### A. GREY WOLF OPTIMIZER (GWO)

GWO is based on the social and behavioral characteristics of grey wolves. The search agents are divided into three classes: Alpha ($\alpha$), Beta ($\beta$), and Delta ($\delta$), which are the agents with the three best solutions, being $\alpha$ the best one. All other agents are termed *Omega* ($\omega$).

Two equations model the gray wolves hunting behavior:

$$D^t_{q,i} = |C \cdot X^t_q - X^t_i| \tag{1a}$$
$$X^t_{D_q} = X^t_q - A \cdot (D^t_{q,i}), \quad q = \alpha,\ \beta\ and\ \delta, \tag{1b}$$

where $X^t_\alpha$, $X^t_\beta$ e $X^t_\delta$ are the positions of the three best solutions and $D^t_{\alpha,i}$, $D^t_{\beta,i}$ and $D^t_{\delta,i}$ represent the distance between them and the agent $X^t_i$, respectively, in the $t$ iteration. The coefficients are $C = 2 \cdot r_1$ and $A = 2a \cdot r_2 - a$, where $a$ is a variable that decreases linearly over the iterations from 2 to 0, so $A$ has random values in the interval [-*a*,*a*] which alternates between exploration ($|A| > 1$) and exploitation ($|A| < 1$) phase. The $r_1$ and $r_2$ are random numbers in the range [0,1].

In a search space the optimal point is unknown; therefore, the three best solutions ($\alpha$, $\beta$, and $\delta$) are used to update the positions of all search agents. Through the equations presented above, the displacement in the iteration $t + 1$ of the search agent $X_i$ is defined by (2), as follows:

$$X^{t+1}_i = \frac{X^t_{D_{\alpha,i}} + X^t_{D_{\beta,i}} + X^t_{D_{\delta,i}}}{3}. \tag{2}$$

### B. MOTH FLAME OPTIMIZATION (MFO)

The MFO meta-heuristic mimics the behavior of moths attracting to a source of artificial light. This attraction is based on a spiral movement, presented by (3):

$$X^{t+1}_{i,l} = D_i \cdot e^{bm} \cdot cos(2\pi m) + Fj, \tag{3}$$

where $X(i, l)$ is $l_{th}$ dimension of the $i_{th}$ search agent, $Fj$ the position of the $j_{th}$ best moth (flame), $b$ is a constant value of 2, $m$ is a random value in the interval $[-1,1]$, and $D_i = |Fj - X_i|$ is the difference between the $j_{th}$ flame and the $i_{th}$ agent. For

$m = -1$ the search agent is close to the flame, if $m = 1$ the agent remains in its place.

To avoid premature convergence of the meta-heuristic into local minimum, the ratio between the flame and the attracted moth starts at 1:1. That is, there is no chance that all moths are attracted to a single flame. If the number of flames is constant, there would be no exploitation of the solution if many regions are being explored. Thus, the number of flames will be updated as follows:

$$n_F = \text{round}\left(n - t * \frac{n-1}{I}\right), \tag{4}$$

where $n$ is the quantity of search agents and $I$ is the maximum number of iterations. The round $(\cdot)$ function returns the integer closest to value in parentheses. The flame updated is based on the sort operation of the moth fitness in two consecutive iterations, in which of $n_F$ best values are chosen like the new flames.

## III. SELF-ADAPTATION APPLIED TO GWO AND MFO

In order to improve the performance of the meta-heuristics GWO and MFO, the adaptive modified GWO (AMGWO) and adaptive MFO (AMFO) are proposed and presented in subsections III-A and III-B.

### A. THE ADAPTIVE MODIFIED GWO (AMGWO)

The AMGWO incorporates two different strategies to improve the meta-heuristic performance: (a) the self-adaptive weighted classes, and (b) a crossing-over approach of the worst search agent.

- *Self-adaptive weighted classes:*

The original GWO updates the position of the search agents by averaging three components, $X_{D_\alpha}$, $X_{D_\beta}$, and $X_{D_\delta}$, which are calculated according to the three best agents named $\alpha$, $\beta$ and $\delta$. It is known that in nature, alpha wolfs are the leaders of the group, and their command is above the others. Therefore, applying the same weight of importance to the three classes would not represent the actual behavior of the pack. On the other hand, according to the occasion, $\beta$ and $\delta$ can assume the command in case of enemy warning or finding prey before $\alpha$.

To overcome these restrictions a new proposal for (2) was elaborated:

$$X_i^{t+1} = \frac{p_\alpha^{t+1} \cdot X_{D_\alpha} + p_\beta^{t+1} \cdot X_{D_\beta} + p_\delta^{t+1} \cdot X_{D_\delta}}{3}, \tag{5}$$

where the values of $p_\alpha$, $p_\beta$ and $p_\delta$ represent the weights of $\alpha$, $\beta$ and $\delta$, respectively, whose values are unique for all search agents at each iteration. However, constant values can not be assigned to these weights since some classes may have more importance than others in some occasions. Thus, based on self-adaptations of JADE [22] parameters, the weights of (5) are adaptive and updated by the following expression:

$$p_q^{t+1} = Mp_q^{t+1} + 0.1 \cdot \text{randn}(0, 1), \quad q = \alpha, \beta, \delta. \tag{6}$$

in which $randn(0, 1)$ is a function that returns a value of the standard normal distribution and $Mp_q$ is the average value of updating the weights, and its value is obtained by:

$$Mp_q^{t+1} = (1-c) \cdot Mp_q^t + c \cdot \text{mean}(GoodP_q), \quad q = \alpha, \beta, \delta, \tag{7}$$

where $c$ is a constant of 0.1 and $GoodP_q$ is the file that stores the value of the weights when $\alpha$, $\beta$ or $\delta$ are updated, i.e., the leaders that have had better fitness than previous iteration. This file starts with five values equal to 1, and over iterations the file is updated by removing the first element inserted to remain with the same size of 5 components.

- *Crossing-over approach of worst search agent:*

Analyzing the search agents during the execution of meta-heuristics it was observed that the agents with the best solution ($\alpha$, $\beta$ e $\delta$) were different from each other in some dimensions near the end of iterations. For a problem with high dimensionality, these variations are relevant in the solution.

To aid tuning of each dimension values, inspired by DE's crossover operation [12], another modification was applied. Therefore, it was proposed to change the particle with the worst solution ($X_{last}$) during each iteration, performing the process of crossing-over the information of leaders, as follows:

$$X_{last,j}^{t+1} = \begin{cases} X_{\alpha,j}^t & \text{if } r \le 1 \\ X_{\beta,j}^t & \text{if } 1 < r \le 2 \\ X_{\delta,j}^t & \text{if } 2 < r \le 3, \end{cases} \tag{8}$$

where $r$ is a random value generated in the interval [0,3] for each dimension $j$. Different from the method applied in [45] our operator is simpler, and we only apply to the worst individual of the swarm. The proposed AMGWO is detailed in Algorithm 1. The computational complexity of AMGWO depends on the number of grey wolves ($n$), the dimensionality of the vector (number of decision variables, $d$), and the maximum number of iterations ($I$). Except for the *calculate* function in line 12 which has a complexity as $O(d)$, we assume a complexity constant ($O(1)$) for all *calculate*, *update*, and *evaluate* steps depicted in lines 9 to 11 and 13, lines 15 to 17, and line 19 of Algorithm 1. Hence, the computational complexity is defined as (9a) and approximated by (9b):

$$O(AMGWO) = O(I \times (O(n \times d) + 3 \times O(1) + O(d))) \tag{9a}$$
$$O(AMGWO) = O(I \cdot n \cdot d + I \cdot d), \tag{9b}$$

where $I$ is the maximum number iterations, $n$ the number of grey wolves, and $d$ is the number of decision variables.

### B. THE ADAPTIVE MFO (AMFO)

The MFO has an operation to simulate the flight of a moth in a spiral movement. This operation is performed by (3) in which one of the exponential parameters is $b$ of value 2.

The adaptive MFO was proposed with a self-adaptation of parameter $b$. That is, each search agent has a particular value of $b$. In this case, the calculation of $b_i$ and its average $Mb_i$

**Algorithm 1** Adaptive Modified Grey Wolf Optimization (AMGWO)

```
INPUT:   Objective Function ($f(\cdot)$)
         Iterations ($I$)
         Number of individuals ($n$)
         Dimension vector ($d$)
OUTPUT:  Best solution ($\boldsymbol{x}_{best}$)
 1: procedure AMGWO
 2: Initialize $\boldsymbol{x}_i^1$ $(i = 1, \ldots, n)$
 3: Initialize $p_q^1 = Mp_q^1 = 1$ $(q = \alpha, \beta, \delta)$
 4: Evaluate $f(\boldsymbol{x}_i^1)$
 5: Update $\boldsymbol{x}_q^1$ $(q = \alpha, \beta, \delta)$
 6: for $t = 1$ to $I$ do
 7:     Update $a$
 8:     for $i = 1$ to $n$ do
 9:         Update $A$ and $C$                          ▷ Eq. 1(a) and 1(b)
10:         Calculate $D_{q,i}^t$ $(q = \alpha, \beta, \delta)$        ▷ Eq. 1(a)
11:         Calculate $X_{D_{q,i}^t}$ $(q = \alpha, \beta, \delta)$       ▷ Eq. 1(b)
12:         Calculate $\boldsymbol{x}_i^{t+1}$                        ▷ Eq. 5
13:         Evaluate $f(\boldsymbol{x}_i^{t+1})$
14:     end for
15:     Update $\boldsymbol{x}_q^{t+1}$ and $GoodP_q$ $(q = \alpha, \beta, \delta)$
16:     Update $Mp_q^{t+1}$ $(q = \alpha, \beta, \delta)$               ▷ Eq. 7
17:     Update $p_q^{t+1}$ $(q = \alpha, \beta, \delta)$                ▷ Eq. 6
18:     for $j = 1$ to $d$ do
19:         Update $\boldsymbol{x}_{last,j}$                          ▷ Eq. 8
20:     end for
21: end for
22: return $\boldsymbol{x}_{best}$
23: end procedure
```

for each agent $i$ is identical to that shown in (6) and (7) of AMGWO:

$$Mb_i^{t+1} = (1 - c) \cdot Mb_i^t + c \cdot \text{mean}(GoodB_i) \quad (10a)$$
$$b_i^{t+1} = M_b^{t+1} + 0.1 \cdot \text{randn}(0, 1), \quad (10b)$$

where $c$ has the value of 0.1, the file $GoodB_i$ starts empty and is filled up during the iterations when the solution of the $i$-th agent is better than that of the previous iteration, up to a maximum size of 5. Upon reaching this value, the first element inserted is removed for the insertion of the new one. The value of $b_i^1$ is 1 and in other iterations follows the system of (10).

The summarized steps of AMFO is shown in Algorithm 2. The sort operation uses the *quicksort* algorithm and is depicted in lines 4 and 14. In line 4 the operation is related to the sort of the total ($n$) number of the individuals. In line 14 the sort includes individuals of previous ($\boldsymbol{x}_n^t$) and current ($\boldsymbol{x}_n^{tC1}$) iterations, but in this case the flames are limited by the first $n_F$ of the sort operation result. In line 8, $m$ is related to (3) and updated by a random function in range of $[-1, 1]$.

The computational complexity of AMFO depends on the number of moths ($n$), the dimensionality of the vector

**Algorithm 2** Adaptive Moth Flame Optimization (AMFO)

```
INPUT:   Objective Function ($f(\cdot)$)
         Iterations ($I$)
         Number of individuals ($n$)
         Dimension vector ($d$)
OUTPUT:  Best solution ($\boldsymbol{x}_{best}$)
 1: procedure AMFO
 2: Initialize $\boldsymbol{x}_i^1$ and $b_i^1 = 1$ $(i = 1, \ldots, n)$
 3: Evaluate $f(\boldsymbol{x}_i^1)$
 4: $F = \text{sort}(\boldsymbol{x}_n^1; n)$
 5: for $t = 1$ to $I$ do
 6:     for $i = 1$ to $n$ do
 7:         for $l = 1$ to $d$ do
 8:             Update $m$
 9:             Calculate $\boldsymbol{x}_{i,l}^{t+1}$                    ▷ Eq. 3
10:         end for
11:         Evaluate $f(\boldsymbol{x}_i^{t+1})$
12:     end for
13:     Update $n_F$                                    ▷ Eq. 4
14:     $F = \text{sort}(\boldsymbol{x}_n^{t+1}, \boldsymbol{x}_n^t; n_F)$
15:     Update $GoodB_i$ $(i = 1, \ldots, n)$
16:     Update $Mb_i^{t+1}$ $(i = 1, \ldots, n)$                ▷ Eq. 10(a)
17:     Update $b_i^{t+1}$ $(i = 1, \ldots, n)$                 ▷ Eq. 10(b)
18: end for
19: return $\boldsymbol{x}_{best}$
20: end procedure
```

(number of decision variables, $d$), the maximum number of iterations ($I$), and the sorting mechanism of flames in each iteration. In our case, the internal sort has a computational complexity of $O(nlogn)$ and $O(n^2)$ for the best and worst cases (of *quicksort*). We assume a complexity of $O(1)$ for all *calculate* and *update* steps depicted in lines 15 to 17, and that the sort is achieved over the $n_F$ elements of the array of flames. Therefore, the computational complexity is defined as (11a) and approximated by (11b):

$$O(AMFO) = O(I \times (O(n \times d) + O({n_F}^2) + 3 \times O(1))) \quad (11a)$$
$$O(AMFO) = O(I \cdot n \cdot d + I \cdot {n_F}^2), \quad (11b)$$

where $I$ is the number iterations, $n$ the number of moths, $d$ is the number of decision variables, and $n_F$ the current number of flames at each interaction.

## IV. NONLINEAR MODEL PREDICTIVE CONTROL PROBLEM DEFINITION

In this paper the meta-heuristics plays the role of an NMPC solver applied in two nonlinear control systems, which are presented in the following.

### A. MODEL-BASED PREDICTIVE CONTROL ALGORITHM

An MPC control strategy is centered on the system model to predict its future behavior over a finite horizon ($N$) based on the current measured and/or estimated states as shown in Fig. 1. It uses a cost function to evaluate the best control

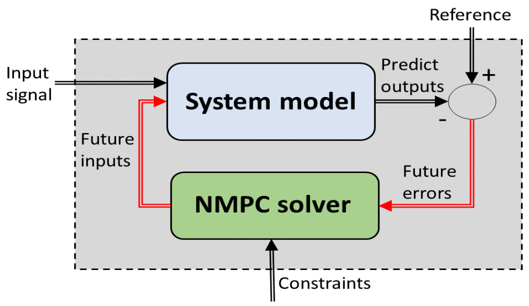


**FIGURE 1.** Block diagram of MPC methodology, *adapted from* [64].

actions based on the input references and state constraints. The strategy consists of four steps which are repeated at each sampling period as listed in Algorithm 3. The NMPC version of the algorithm uses the same steps using the nonlinear system model.

**Algorithm 3** MPC Strategy

```
1: for decision instant $k$ do
2:     Obtain controlled variable measurements;
3:     Calculate the sequence of future actions that minimize the system cost function;
4:     Apply the first control action of the optimal sequence in time interval $[k, k+1]$;
5:     Proceed to the next instant $k = k + 1$;
6: end for
```

The NMPC was applied on two nonlinear dynamic systems. Such systems are presented below.

### B. SINGLE INVERTED PENDULUM CONTROL

The first control problem aims to perform a constrained inverted pendulum swing-up while also bringing it to a reference cart position ($x = 0$). In summary, the acceleration of the cart and the angular acceleration of the pendulum are respectively:

$$\ddot{x} = \frac{1}{M+m}[u - b\dot{x} - ml\ddot{\theta}c(\theta) + ml\dot{\theta}^2 s(\theta)] \quad (12)$$

$$\ddot{\theta} = \frac{3}{4ml^2}[mgl \cdot s(\theta) - ml\ddot{x}c(\theta) - h\dot{\theta}], \quad (13)$$

where $M$ is the mass of the cart, $m$ is the uniformly distributed mass of the pendulum, and $2l$ is its length. In this case, $b$ is the friction coefficient of the cart with the surface, $h$ is the friction coefficient of rotation, $u$ is the force applied to the cart, $\theta$ is the angle between the normal force and the pendulum, $s(\theta)$ is a sin of $\theta$, and $c(\theta)$ is cosine of $\theta$. Finally, $x$ is the horizontal displacement of the cart.

The system is discretized by a 4th order Runge-Kutta method. The complete mathematical modeling can be consulted in [65]. Denoting the states of the system as a vector $X = [x\,\dot{x}\,\theta\,\dot{\theta}]^T$, the cost function that the NMPC-solver seeks to minimize is shown in (14) [66]:

$$u_{opt} = \text{argmin}\left[\sum_{i=1}^{N-1}\left(\sum_{j=1}^{4} Q_e(X(j,i)) \cdot (X(j,i) - X_{ref}(j,i))^2 \cdot Q(j) + R \cdot (u(i) - u_{ref}(i))^2\right) + \sum_{j=1}^{4} Q_e(X(j,N)) \cdot (X(j,N) - X_{ss}(j))^2 Q_f(j)\right], \quad (14)$$

where $X_{ss} = X_{ref} = (x = 0, \dot{x} = 0, \theta = 0°, \dot{\theta} = 0°)$ is the desired steady-state which is equal to the reference, $Q_e = (10^4)$ is a penalty applied to obey the $x$ state constraint and is applied whenever $x > 0.5m$ or $x < -0.5m$. $Q(j)$ and $Q_f(j)$ are the state trajectory error penalties $j$ in horizons 1 to $N-1$ and final $N$, respectively.

### C. SATELLITE ATTITUDE CONTROL

The second control problem has the objective of controlling the attitude of a satellite in the three rotation axes of a simulated test platform [67]. The element responsible for the orientation of each axis is a *reaction wheel* which is connected to a motor that induces a speed and, by the principle of conservation of angular momentum, a torque of the same intensity and opposite direction applied to the body [68].

The attitude of the simulator is defined as the *relative orientation* between the inertial reference system $F_i$ $(i_1, i_2, i_3)$ and the body reference system $F_b$, relative to the moving part of the platform, but having the same center as $F_i$.

The problem is to take the attitude of the body, initially $(0°, 0°, 0°)$, to a reference $(\theta_1, \theta_2, \theta_3)$. The control action is numbered by $n_u$ and represents the acceleration of the motor acting on each reaction wheel.

The mathematical model of the platform is described by (15) [67]:

$$\begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} = \begin{bmatrix} 0 & s_3/c_2 & c_3/c_2 \\ 0 & c_3 & -s_3 \\ 1 & s_3 s_2/c_2 & c_3 s_2/c_2 \end{bmatrix} \cdot \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}, \quad (15)$$

where the values of $\theta_i$ are the Euler angles describing the attitude of the simulator as the relative orientation between the inertial frame $F_i$ and the reference fixed to the body $F_b$, $s_i$ and $c_i$ are abbreviations for the sine and cosine of $\theta_i$ and $\omega_i$ are the angular velocities of the reference $F_b$ in relation to $F_i$. Therefore, the system can be represented in a state space:

$$\dot{X} = f(X, u, A, B), \quad (16)$$

where the state vector is $X = (\theta_1\ \theta_2\ \theta_3\ \omega_1\ \omega_2\ \omega_3)^T$ and the control action is $u = (\dot{\Omega}_1\ \dot{\Omega}_2\ \dot{\Omega}_3)^T$; in which $\dot{\Omega}_1$, $\dot{\Omega}_2$ and $\dot{\Omega}_3$ are the acceleration of the reaction wheels. Therefore, $\Omega_1$, $\Omega_2$ and $\Omega_3$ are the velocities of the reaction wheels. The matrices $A$ and $B$ are described in [68], and the former depends on the variable values of $\Omega_{1,2,3}$ which makes the system nonlinear.

It is important to mention that the formulation herein depicted also encompasses models with time delay [69]–[71], and this can be achieved by the proper definition of $u(t)$.

To achieve the overall control system two strategies have been used:

#### 1) FILTERED REFERENCE

The first strategy uses a filtered reference as proposed in [68], and represented by (17):

$$y_i(t) = \theta_i(1 - e^{\frac{-3\tau_s}{t_{ref}}}), \tag{17}$$

where $\theta_i$ is the amplitude of the reference signal, $\tau_s$ is the sampling period, and $t_{ref}$ the response time for the system to reach 95% of the amplitude $\theta_i$. The application of this input signal is intended to reduce the overshoot of the output signal.

#### 2) EXPONENTIAL PARAMETERIZATION

The second strategy is an exponential parameterization that is based on exponential terms to obtain a candidate command sequence of the control signal. The system input ($\mathbf{u}$) must 'see' the prediction horizon ($N$), that is, the higher N the greater the complexity to define $\mathbf{u}_{opt}$. To decrease the size of the decision variable and decouple it from the $N$ value, the parameterization of $\mathbf{u}$ is performed. It is represented by the sum of exponentials to model the signal through two coefficients:

$$u_n(t + i\tau_s) = \sum_{l=1}^{n_e^n} \left[ e^{\frac{-\lambda_n}{(l-1)\alpha+1}} \right] \cdot p_l^{(n)}, \tag{18}$$

where $t \in [(k-1)\tau_s, k\tau_s[$, $n$ is the n-*th* actuator of system, $i \in \{0, \cdots, N-1\}$, $\lambda_n > 0$ and $\alpha > 1$ are tuning parameters. The $n_e^n$ is the number of exponential terms chosen for n-*th* actuator. Thus, the new set of parameters $p$ that may be found is:

$$p = \begin{pmatrix} p^{(1)} \\ \vdots \\ p^{(n_u)} \end{pmatrix}. \tag{19}$$

More details about the exponential parameterization are found in [72]. The parameters $p^n$ are obtained by a cost function to be optimized by the NMPC solver:

$$p_{opt} = \text{argmin} \left[ \sum_{i=1}^{N-1} \left( \sum_{j=1}^{6} (\theta(j,i) - \theta_{ref}(j,i))^2 \right) + \sum_{j=1}^{6} Q_f(j) \cdot (\theta(j,N) - \theta_{ss}(j))^2 \right], \tag{20}$$

where $\theta_{ss} = (\theta_1 = 50°, \theta_2 = -30°, \theta_3 = 60°, \omega_1 = 0, \omega_2 = 0, \omega_3 = 0)$ are the reference values of the states, $Q(j)$ and $Q_f(j)$ are the same penalties for pendulum control. After the solution of (20) only the first control signal $u_n(p_{opt}, 1)$ is applied to the system in $[t, t + \tau_s]$ and in the next instants the whole methodology is applied again.

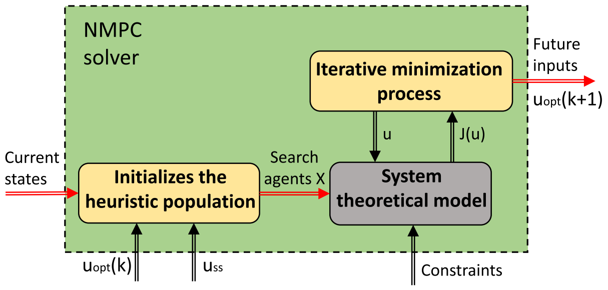


**FIGURE 2.** Internal solver process performed by the bioinspired meta-heuristic.

### D. NMPC SOLVER

The NMPC solver process is performed at each sample time, and it is divided by three main blocks shown in Fig. 2. Those lighter color refer to meta-heuristics tasks. The initialization of the heuristic population is composed of two methods: the previous best agent and the steady-state agent.

- *Previous best agent*: the optimized result obtained for the instant $k$ ($u_{opt}(k)$) is used as one of the search agents of the initial population of the meta-heuristic in instant $k+1$. This method was proposed by [73], being suitable since the pendulum and the satellite are slow dynamic systems, and their states do not suffer abrupt variations from one sampling instant to another.
- *Steady-state agent*: the initialization of one search agent is done with the value of the steady-state control action (in both of these control problems $\mathbf{u}_{ss} = 0$ for all actuators) since the search agents tend to converge to $\mathbf{u}_{ss}$ at the end of the control process [66].

The iterative minimization process is based on each meta-heuristic optimization features that over a number of iterations find the optimal control action ($u_{opt}$) that minimizes the cost function $J(u)$ provided by the system's model. After that, the first control element ($u_{opt}(k+1)(1)$) is applied to the system, and a new optimization process is computed in the next sampling interval.

## V. RESULTS

This section shows the experimental aspects used in this research and our results obtained on the two benchmarks (*single inverted pendulum* and *satellite attitude*), by using the metaheuristics proposed here. In this sense, the experimental setup is shown in subsection V-A, the nonparametric analysis and test results are shown in subsection V-B, the cost function convergence is shown in subsection V-C and the dynamic of adapted parameters are shown in subsection V-D.

### A. EXPERIMENTAL SETUP

The definition of problems in the case of the pendulum are the sampling intervals where the control action needs to be computed, and for the satellite, the median of these instants. In this case, we have the following sample strategies:

- *Single inverted pendulum*: Each sampling interval is defined as an individual problem. The simulation duration is 20 seconds long, with a sampling interval of $\tau_s = 0.1s$ resulting in 200 optimization problems. The first fourteen seconds were discarded since all meta-heuristics have similar performance in this period. Furthermore, the system is far from reaching the reference states and, consequently, the convergence, which represents a high-cost function value. Thus, 50 problems were defined, being each decision instant between 14 and 18.9 seconds of simulation.

- *Satellite*: The satellite system was simulated for 100 seconds, thus setting 1000 decision instants for $\tau_s = 0.1s$. The 50 problems were defined by extracting the average of every 16 consecutive sampling times until $t = 80$ seconds.

The results are defined by the *median* of 51 runs of each meta-heuristic. The medians are used since the averages are likely to allow good performance in a data-set to compensate for the overall poor performance of the meta-heuristic [59]. This guidance is followed by [60], which recommends the use of medians in the comparison of multiple meta-heuristics.

For each control problem, specific simulation parameters were defined. For control of inverted pendulum, the parameters defined by [66] were used: (i) 10 particles and 30 iterations in meta-heuristics; (ii) the pendulum parameters being $m = 7.3\ kg$, $M = 14.6\ kg$, $l = 1.2\ m$, $b = 14.6\ kg/s$ and $h = 0.0136\ kg \cdot m^2/s$; and (iii) the NMPC parameters being $N = 20$, $N_u = 20$, $\tau_s = 0.1\ s$, $Q = [10^4\ 1\ 10^4\ 1]$, $Q_f = [10^3\ 10^3\ 10^3\ 10^3]$ and $R = 1$.

In the satellite control, the parameters are based on [74]: (i) 4 particles and 15 iterations in meta-heuristics; (ii) the satellite parameters being $I_\omega = 1.8 \cdot 10^{-3}\ kg \cdot m^2$, $I_{11} = I_{22} = I_{33} = 1.17 \cdot 10^{-3}\ kg \cdot m^2$; and (iii) the NMPC parameters being $N = 75$, $N_u = 75$, $\tau_s = 0.1\ s$, $Q = [10^2\ 10^2 10^2\ 1\ 1\ 1]$, $Q_f = [1]$, $\lambda = 0.1$, $q = 8.0$, $\delta_u = 1\ rad/s$, $t_{ref} = 30\ s$.

## B. STATISTICAL ANALYSIS AND TEST

In multiple comparisons of meta-heuristics, there are no significance results when an inadequate amount of problems are chosen. Thus, the relation of appropriate number of problems ($n_p$) between the quantity of compared meta-heuristic ($n_m$) is defined by $2 \cdot n_m \leq n_p < 8 \cdot n_m$ [62]. Therefore, in our study (with 7 meta-heuristics) the number of problems are limited from $14 \leq n_p < 56$. In this context, the Shapiro-Wilk normality test was applied. Once the non-normal distribution was verified in data of both case studies, the performed comparison was obtained by nonparametric methods. Such methods are based on the Friedman test [75] and the Shaffer post-hoc procedure [76], highly recommended for multiple meta-heuristic comparison in multiple problems [60]. The nonparametric tests were performed by KEEL (*Knowledge Extraction based on Evolutionary Learning*) [77], [78].

The ranking of the Friedman test (Table. 1) shows that there are differences in performance for at least two metaheuristics in both control systems, since the *p-values* $< 0.05$. These differences have been found by the Shaffer post-hoc test as presented in Table. 2

**TABLE 1.** Ranking of metaheuristics.

| Pendulum | | Satellite | |
|---|---|---|---|
| Metaheuristic | *Ranking* | Metaheuristic | *Ranking* |
| **AMFO** | 1.00 | **AMGWO** | 2.40 |
| MFO | 2.58 | JADE | 2.48 |
| **AMGWO** | 2.80 | LSHADE | 3.28 |
| PSO | 4.16 | **AMFO** | 4.14 |
| GWO | 4.62 | MFO | 4.40 |
| LSHADE | 5.90 | GWO | 5.22 |
| JADE | 6.94 | PSO | 6.08 |
| *p-value* | $< 0.001$ | *p-value* | $< 0.001$ |

**TABLE 2.** Shaffer post-hoc test.

| i | Pendulum | $p_{Shaf}$ | Satellite | $p_{Shaf}$ |
|---|---|---|---|---|
| 1 | **AMFO** vs .JADE | 0.00 | **AMGWO** vs .PSO | 0.00 |
| 2 | **AMFO** vs .LSHADE | 0.00 | JADE vs .PSO | 0.00 |
| 3 | JADE vs .MFO | 0.00 | **AMGWO** vs .GWO | 0.00 |
| 4 | **AMGWO** vs .JADE | 0.00 | LSHADE vs .PSO | 0.00 |
| 5 | **AMFO** vs .GWO | 0.00 | JADE vs .GWO | 0.00 |
| 6 | MFO vs .LSHADE | 0.00 | **AMGWO** vs .MFO | 0.00 |
| 7 | **AMFO** vs .PSO | 0.00 | LSHADE vs .GWO | 0.00 |
| 8 | **AMGWO** vs .LSHADE | 0.00 | **AMFO** vs .PSO | 0.00 |
| 9 | JADE vs .PSO | 0.00 | JADE vs .MFO | 0.00 |
| 10 | JADE vs .GWO | 0.00 | **AMGWO** vs **.AMFO** | 0.00 |
| 11 | MFO vs .GWO | 0.00 | MFO vs .PSO | 0.00 |
| 12 | **AMGWO** vs .GWO | 0.00 | **AMFO** vs .JADE | 0.00 |
| 13 | **AMGWO** vs **.AMFO** | 0.00 | MFO vs .LSHADE | 0.08 |
| 14 | LSHADE vs .PSO | 0.00 | **AMFO** vs .GWO | 0.08 |
| 15 | **AMFO** vs .MFO | 0.00 | **AMGWO** vs .LSHADE | 0.29 |
| 16 | MFO vs .PSO | 0.00 | **AMFO** vs .LSHADE | 0.29 |
| 17 | **AMGWO** vs .PSO | 0.00 | GWO vs .PSO | 0.29 |
| 18 | LSHADE vs .GWO | 0.01 | MFO vs .GWO | 0.29 |
| 19 | JADE vs .LSHADE | 0.04 | JADE vs .LSHADE | 0.29 |
| 20 | GWO vs .PSO | 0.57 | **AMFO** vs .MFO | 1.00 |
| 21 | **AMGWO**vs .MFO | 0.61 | **AMGWO** vs .JADE | 1.00 |

The post-hoc test shows which metaheuristics have had different performance when $p_{Shaf} < 0.05$. Thus, the hypotheses 1 to 19 from the pendulum control problem and the hypotheses 1 to 12 indicates that the metaheuristics, in each pair test, had different performance from each other. The results shows, therefore, the proposed metaheuristics AMGWO and AMFO have improved the performance of GWO (hypothesis 12) and MFO (hypothesis 15), respectively, for the pendulum control. Conversely, for the satellite attitude control, only AMGWO has improved GWO (hypothesis 3), since the hypothesis 20 (AMFO vs. MFO) resulted in $p_{Shaf} > 0.05$.

## C. COST FUNCTION AND CONTROL ACTION

The cost function shows how the meta-heuristic performed on the minimization task as an NMPC solver over the expended time. The cost function convergence for the single pendulum control is presented in Fig. 3.

The controller used the first twelve seconds to swing-up the pendulum, which results in a non-descendent cost function in this period, as shown in detail in Fig. 3. Because of that, the problems were defined after the 14th second.

The proposed meta-heuristics showed a more pronounced cost function convergence than its originals in the descendent period. The AMFO showed the fastest convergence for single pendulum control. Besides, this meta-heuristics presented

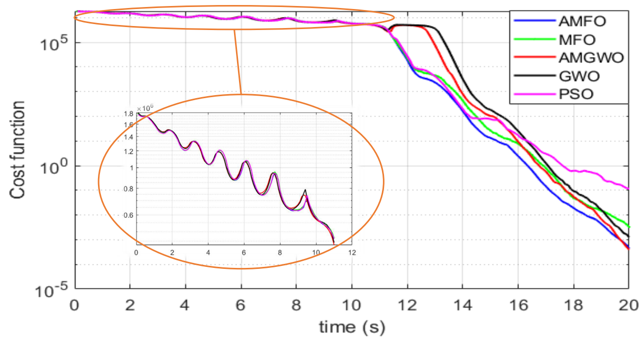


**FIGURE 3.** Cost function convergence for single pendulum control.

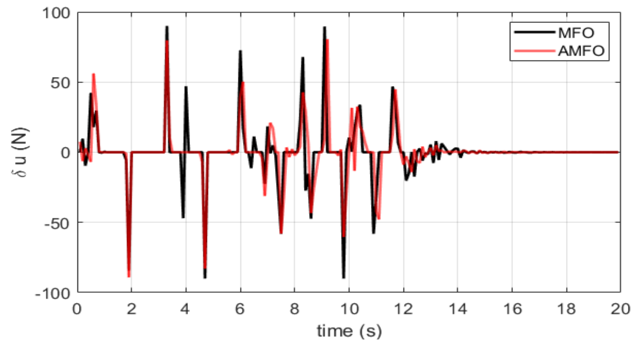


**FIGURE 4.** Control action variation for single pendulum control.

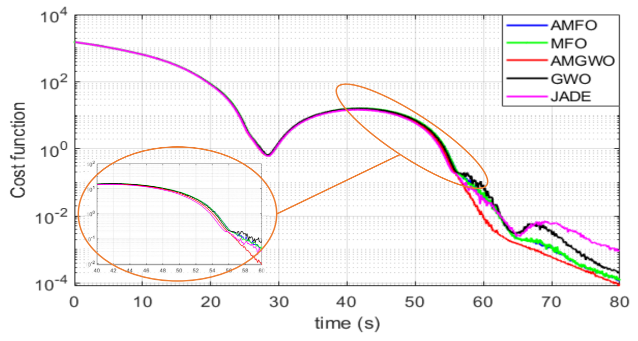


**FIGURE 5.** Cost function convergence for attitude satellite control.

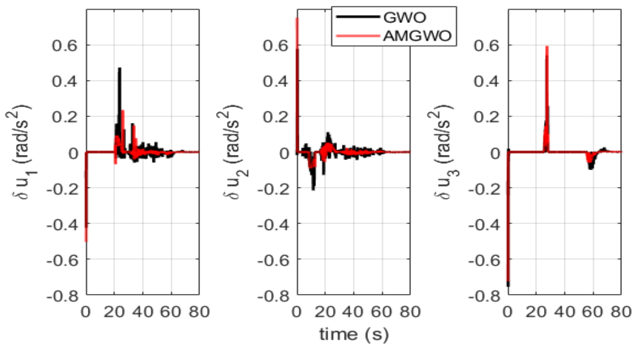


**FIGURE 6.** Control action variation for attitude satellite control.

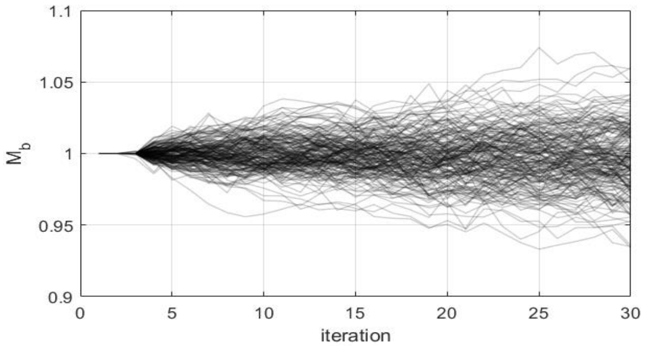


**FIGURE 7.** Best Mb values over the iteration for AMFO in single pendulum control.

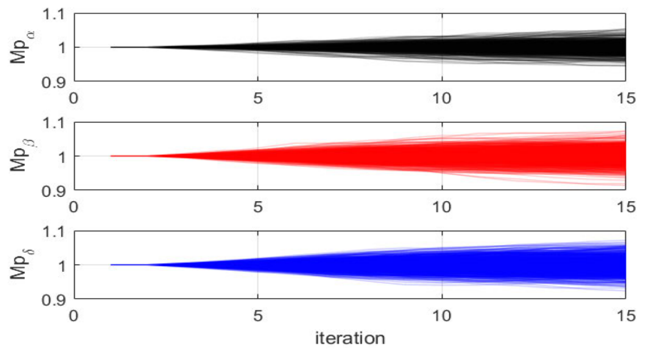


**FIGURE 8.** Mp values over the iteration for AMGWO in attitude satellite control.

lower control action variation ($\delta u$) than MFO, as shown in Fig. 4.

The same analysis was made for the satellite attitude control system, where the cost function convergence is shown in Fig. 5. The JADE and AMGWO demonstrated to have lower values than other meta-heuristics in the first 55 seconds of simulation. The AMGWO presented the best performance in the final simulation period.

For all three control actions in the satellite system, the AMGWO presented lower control variations ($\delta u_1$, $\delta u_2$ and $\delta u_3$) than the GWO, as shown in Fig. 6.

### D. SELF-ADAPTATION CHARACTERISTICS

To validate the use of self-adaptive parameters applied to meta-heuristics-based NMPC solvers, the following results can be observed. Fig. 7 shows the $Mb$ value for the best search agent in each iteration of the AMFO applied to the pendulum control. Fig. 8 shows the Mp values ($Mp_\alpha$, $Mp_\beta$ and $Mp_\delta$) in each iteration of the satellite attitude control when the AMGWO was applied. In these figures, each line represents a sample instant of the simulation.

These results reinforce the importance of applying parameter adaptation strategies since control systems like the pendulum and satellite require different tuning at each sampling interval.

## VI. RESULT ANALYSIS

In the present study, we proposed two self-adaptation meta-heuristics, named Adaptive Modified GWO and Adaptive MFO, applied in two NMPC control systems. We hypothesized that simple self-adaptation methods can improve the meta-heuristic performance in the minimization task of the NMPC solver. Then, we compared the performance of the proposed meta-heuristics with their respective original GWO and MFO algorithms, a classical PSO, and two self-adaptation versions of the Differential Evolution, JADE and LSHADE.

The Friedman test (Table. 1) suggested that the modifications and self-adaptation applied to GWO and MFO have improved their original performance in both nonlinear control systems since the AMGWO and AMFO have had the best ranking than GWO and MFO. The Shaffer post-hoc test (Table. 2) confirmed that AMFO and AMGWO had better performance than MFO and GWO in the single pendulum control, but in the satellite attitude control, only the AMGWO has shown improvement over the GWO, since the AMFO resulted in statistically similar performance with the MFO.

These performance improvements were confirmed by the lowest and fastest cost function convergence of the proposed meta-heuristics, observed mainly in the single pendulum control (Fig. 3). The novel mechanisms proposed in AMGWO, i.e., self-adaptive weighted classes and the crossing-over approach in the worst search agent, have confirmed the hypotheses. The results have shown better exploitation behavior and a speed-up in convergence time in both nonlinear control systems. In addition, the AMFO and AMGWO have established less variation in the control action than their original algorithms, in the pendulum (Fig. 4) and satellite (Fig. 6) controls, respectively. In practice, the smaller the control variation signals, the lower the energy expenditure, which guarantees longer durability of the actuators and improve their performance over time.

The NMPC process presented different values of variables used to calculate the adaptive parameters, Mb (Fig. 7) and Mp (Fig. 8), in each sampling time. This demonstrates that such parameters must be adjusted according to the system dynamics, since each control instant presents different system states and, consequently, a different minimization task. Thus, the advantage of applying adaptation techniques in meta-heuristics is that their values are automatically addressed according to the current state of the problem, given that each case study has its characteristics and singularities. Furthermore, meta-heuristics capable of adjusting to the problem may achieve better results than those that do not use parameter control strategies.

Despite the good results obtained by the AMGWO and AMFO approaches, there are limitations to be considered, given that although they can be applied to any optimization problem, in general, meta-heuristics do not necessarily guarantee the optimal solution (global minimum/maximum). Otherwise, taking into account the no-free-lunch theorem their behavior need to be evaluated in other applications.

## VII. CONCLUSIONS AND FUTURE WORKS

The results of this study are relevant in suggesting that a simple self-adaptive parameter technique can improve the meta-heuristic performance when applied as an NMPC solver. The statistical and the cost function analysis showed that AMFO and AMGWO obtained better performance than their original meta-heuristics in the single pendulum and satellite attitude control problems, respectively. In addition, the proposed methods can reduce the variation of control signal implying in more energy efficiency and durability of the motor drive.

Future works encourage us to test optimizers developed for real-time controller environments and to verify the ability to handle noisy measures such as proposed in [68].

## REFERENCES

[1] J. M. Maciejowski, *Predictive Control With Constraints*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[2] G. Torrisi, S. Grammatico, R. S. Smith, and M. Morari, "A projected gradient and constraint linearization method for nonlinear model predictive control," *SIAM J. Control Optim.*, vol. 56, no. 3, pp. 1968–1999, Jan. 2018.

[3] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen, "A software framework for embedded nonlinear model predictive control using a gradient-based augmented lagrangian approach (GRAMPC)," *Optim. Eng.*, vol. 20, no. 3, pp. 769–809, Jan. 2019.

[4] Z. Liu, L. Xie, A. Bemporad, and S. Lu, "Fast linear parameter varying model predictive control of buck DC-DC converters based on FPGA," *IEEE Access*, vol. 6, pp. 52434–52446, 2018.

[5] A. Mozaffari and N. L. Azad, "Empirical investigation and analysis of the computational potentials of bio-inspired nonlinear model predictive controllers: Success and challenges," *Int. J. Bio-Inspired Comput.*, vol. 9, no. 1, pp. 19–34, 2017.

[6] A. K. Kar, "Bio inspired computing—A review of algorithms and scope of applications," *Expert Syst. Appl.*, vol. 59, pp. 20–32, Oct. 2016.

[7] X.-H. Chang, J. Xiong, and J. H. Park, "Fuzzy robust dynamic output feedback control of nonlinear systems with linear fractional parametric uncertainties," *Appl. Math. Comput.*, vol. 291, pp. 213–225, Dec. 2016.

[8] L. Ma, X. Huo, X. Zhao, and G. Zong, "Adaptive fuzzy tracking control for a class of uncertain switched nonlinear systems with multiple constraints: A small-gain approach," *Int. J. Fuzzy Syst.*, vol. 21, no. 8, pp. 2609–2624, Nov. 2019.

[9] S. Desale, A. Rasool, S. Andhale, and P. Rane, "Heuristic and meta-heuristic algorithms and their relevance to the real world: A survey," *Int. J. Comput. Eng. Res. Trends*, vol. 351, no. 5, pp. 2349–7084, 2015.

[10] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. London, U.K.: Oxford Univ. Press, 1997.

[11] M. Mitchell, "An introduction to genetic algorithms," *Sadhana*, vol. 24, nos. 4–5, pp. 293–315, Aug. 1999.

[12] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Nov./Dec. 1995, pp. 1942–1948.

[14] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

[15] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[16] G. G. Wang, S. Deb, X. Z. Gao, and L. D. S. Coelho, "A new metaheuristic optimisation algorithm motivated by elephant herding behaviour," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 6, p. 394, 2016.

[17] S. Mirjalili, "Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[18] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[19] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[20] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Parameter control in evolutionary algorithms: Trends and challenges," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 167–187, Apr. 2015.

[21] S. Gao, Y. Gao, Y. Zhang, and L. Xu, "Multi-strategy adaptive cuckoo search algorithm," *IEEE Access*, vol. 7, pp. 137642–137655, 2019.

[22] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 945–958, Oct. 2009.

[23] A. Gálvez and A. Iglesias, "New memetic self-adaptive firefly algorithm for continuous optimisation," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 5, pp. 300–317, Jan. 2016.

[24] H. Wang, X. Zhou, H. Sun, X. Yu, J. Zhao, H. Zhang, and L. Cui, "Firefly algorithm with adaptive control parameters," *Soft Comput.*, vol. 21, no. 17, pp. 5091–5102, Sep. 2017.

[25] J. Wu, R. Nan, and L. Chen, "Improved salp swarm algorithm based on weight factor and adaptive mutation," *J. Experim. Theor. Artif. Intell.*, vol. 31, no. 3, pp. 493–515, Feb. 2019.

[26] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2014, pp. 1658–1665.

[27] S. Gupta and K. Deep, "Cauchy grey wolf optimiser for continuous optimisation problems," *J. Experim. Theor. Artif. Intell.*, vol. 30, no. 6, pp. 1051–1075, Aug. 2018.

[28] S. Gupta and K. Deep, "A novel random walk grey wolf optimizer," *Swarm Evol. Comput.*, vol. 44, pp. 101–112, Feb. 2019.

[29] Z.-J. Teng, J.-L. Lv, and L.-W. Guo, "An improved hybrid grey wolf optimization algorithm," *Soft Comput.*, vol. 23, no. 15, pp. 6617–6631, Aug. 2019.

[30] M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *J. Comput. Design Eng.*, vol. 5, no. 4, pp. 458–472, Oct. 2018.

[31] A. A. Heidari and P. Pahlavani, "An efficient modified grey wolf optimizer with Lévy flight for optimization tasks," *Appl. Soft Comput.*, vol. 60, pp. 115–134, Nov. 2017.

[32] L. Rodríguez, O. Castillo, J. Soria, P. Melin, F. Valdez, C. I. Gonzalez, G. E. Martinez, and J. Soto, "A fuzzy hierarchical operator in the grey wolf optimizer algorithm," *Appl. Soft Comput.*, vol. 57, pp. 315–328, Aug. 2017.

[33] W. Long, T. Wu, S. Cai, X. Liang, J. Jiao, and M. Xu, "A novel grey wolf optimizer algorithm with refraction learning," *IEEE Access*, vol. 7, pp. 57805–57819, 2019.

[34] L. Huang, B. Yang, X. Zhang, L. Yin, T. Yu, and Z. Fang, "Optimal power tracking of doubly fed induction generator-based wind turbine using swarm moth–flame optimizer," *Trans. Inst. Meas. Control*, vol. 41, no. 6, pp. 1491–1503, Apr. 2019.

[35] G.-Q. Lin, L.-L. Li, M.-L. Tseng, H.-M. Liu, D.-D. Yuan, and R. R. Tan, "An improved moth-flame optimization algorithm for support vector machine prediction of photovoltaic power generation," *J. Cleaner Prod.*, vol. 253, Apr. 2020, Art. no. 119966.

[36] L. Yue, R. Yang, J. Zuo, H. Luo, and Q. Li, "Air target threat assessment based on improved moth flame optimization-gray neural network model," *Math. Problems Eng.*, vol. 2019, pp. 1–14, Oct. 2019.

[37] A. E. Hassanien, T. Gaber, U. Mokhtar, and H. Hefny, "An improved moth flame optimization algorithm based on rough sets for tomato diseases detection," *Comput. Electron. Agricult.*, vol. 136, pp. 86–96, Apr. 2017.

[38] H. Sheng, C. Li, H. Wang, Z. Yan, Y. Xiong, Z. Cao, and Q. Kuang, "Parameters extraction of photovoltaic models using an improved moth-flame optimization," *Energies*, vol. 12, no. 18, p. 3527, Sep. 2019.

[39] C. Li, Z. Niu, Z. Song, B. Li, J. Fan, and P. X. Liu, "A double evolutionary learning moth-flame optimization for real-parameter global optimization problems," *IEEE Access*, vol. 6, pp. 76700–76727, 2018.

[40] D. Pelusi, R. Mascella, L. Tallini, J. Nayak, B. Naik, and Y. Deng, "An improved moth-flame optimization algorithm with hybrid search phase," *Knowl.-Based Syst.*, vol. 191, Mar. 2020, Art. no. 105277.

[41] M. A. Taher, S. Kamel, F. Jurado, and M. Ebeed, "An improved moth-flame optimization algorithm for solving optimal power flow problem," *Int. Trans. Electr. Energy Syst.*, vol. 29, no. 3, p. e2743, Oct. 2018.

[42] N. Deotale, U. Kolekar, and A. Kondelwar, "Optimal transmit antenna selection for LTE system using self-adaptive grey wolf optimization," *Multiagent Grid Syst.*, vol. 14, no. 1, pp. 67–82, Apr. 2018.

[43] S. Koppu and V. M. Viswanatham, "Medical image security enhancement using two dimensional chaotic mapping optimized by self-adaptive grey wolf algorithm," *Evol. Intell.*, vol. 11, nos. 1–2, pp. 53–71, Aug. 2018.

[44] H. Jia, J. Ma, and W. Song, "Multilevel thresholding segmentation for color image using modified moth-flame optimization," *IEEE Access*, vol. 7, pp. 44097–44134, 2019.

[45] I. Gölcük and F. B. Ozsoydan, "Evolutionary and adaptive inheritance enhanced Grey Wolf optimization algorithm for binary domains," *Knowl.-Based Syst.*, vol. 194, Apr. 2020, Art. no. 105586.

[46] G. H. Negri, M. S. M. Cavalca, and R. S. Parpinelli, "Model-based predictive control using differential evolution applied to a pressure system," *IEEE Latin Amer. Trans.*, vol. 14, no. 1, pp. 89–95, Jan. 2016.

[47] G. H. Negri, V. H. B. Preuss, M. S. M. Cavalca, and J. de Oliveira, "Differential evolution optimization applied in multivariate nonlinear model-based predictive control," in *Proc. Latin Amer. Congr. Comput. Intell. (LA-CCI)*, Oct. 2015, pp. 1–6.

[48] S. S. Kaddah, K. M. Abo-Al-Ez, and T. F. Megahed, "Application of nonlinear model predictive control based on swarm optimization in power systems optimal operation with wind resources," *Electr. Power Syst. Res.*, vol. 143, pp. 415–430, Feb. 2017.

[49] F. Xu, H. Chen, X. Gong, and Q. Mei, "Fast nonlinear model predictive control on FPGA using particle swarm optimization," *IEEE Trans. Ind. Electron.*, vol. 63, no. 1, pp. 310–321, Jan. 2016.

[50] Z. Cai, H. Zhou, J. Zhao, K. Wu, and Y. Wang, "Formation control of multiple unmanned aerial vehicles by event-triggered distributed model predictive control," *IEEE Access*, vol. 6, pp. 55614–55627, 2018.

[51] L. Xiao, M. Xu, Y. Chen, and Y. Chen, "Hybrid grey wolf optimization nonlinear model predictive control for aircraft engines based on an elastic BP neural network," *Appl. Sci.*, vol. 9, no. 6, p. 1254, Mar. 2019.

[52] P. Yao, H. Wang, and H. Ji, "Multi-UAVs tracking target in urban environment by model predictive control and improved grey wolf optimizer," *Aerosp. Sci. Technol.*, vol. 55, pp. 131–143, Aug. 2016.

[53] B. Han, X. Kong, Z. Zhang, and L. Zhou, "Neural network model predictive control optimisation for large wind turbines," *IET Gener., Transmiss. Distrib.*, vol. 11, no. 14, pp. 3491–3498, Sep. 2017.

[54] Y. Lu, D. Li, Z. Xu, and Y. Xi, "Convergence analysis and digital implementation of a discrete-time neural network for model predictive control," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7035–7045, Dec. 2014.

[55] H. Xiao and C. L. P. Chen, "Leader-follower consensus multi-robot formation control using neurodynamic-optimization-based nonlinear model predictive control," *IEEE Access*, vol. 7, pp. 43581–43590, 2019.

[56] B. Zhang, X. Sun, S. Liu, and X. Deng, "Adaptive differential evolution-based distributed model predictive control for multi-UAV formation flight," *Int. J. Aeronaut. Space Sci.*, vol. 21, pp. 1–11, Oct. 2019.

[57] G. Hou, L. Gong, Z. Yang, and J. Zhang, "Multi-objective economic model predictive control for gas turbine system based on quantum simultaneous whale optimization algorithm," *Energy Convers. Manage.*, vol. 207, Mar. 2020, Art. no. 112498.

[58] Y. C. Lin, D.-D. Chen, M.-S. Chen, X.-M. Chen, and J. Li, "A precise BP neural network-based online model predictive control strategy for die forging hydraulic press machine," *Neural Comput. Appl.*, vol. 29, no. 9, pp. 585–596, May 2018.

[59] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[60] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[61] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, p. 617, Dec. 2009.

[62] S. García, A. Fernández, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability," *Soft Comput.*, vol. 13, no. 10, pp. 959–977, Dec. 2008.

[63] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Int. Conf. Evol. Comput. World Congr. Comput. Intell.*, Anchorage, AK, USA, May 1998, pp. 69–73.

[64] E. F. Camacho and C. Bordons, *Model Predictive Control in the Process Industry*. London, U.K.: Springer, 1995.

[65] A. Alaniz, "Model predictive control with application to real-time hardware and guided parafoil," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2004.

[66] R. C. Sampaio, "Arquiteturas de hardware para aceleração de algoritmos de controle preditivo não-linear," Ph.D. dissertation, Dept. Engenharia Mecânica, Univ. Brasília, Brasília, Brazil, 2018.

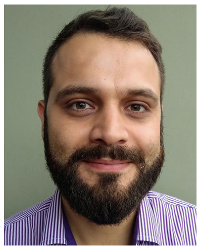
[67] R. G. Gonzales, "Utilização dos métodos SDRE e filtro de Kalman para o controle de atitude de simuladores de satélites," M.S. thesis, Instituto Nacional Pesquisas Espaciais, São José dos Campos, Brazil, 2009.

[68] R. Rodrigues, A. Murilo, R. V. Lopes, and L. C. G. D. Souza, "Hardware in the loop simulation for model predictive control applied to satellite attitude control," *IEEE Access*, vol. 7, pp. 157401–157416, 2019.

[69] B. Wang, W. Chen, B. Zhang, and Y. Zhao, ''Regulation cooperative control for heterogeneous uncertain chaotic systems with time delay: A synchronization errors estimation framework,'' *Automatica*, vol. 108, Oct. 2019, Art. no. 108486.
[70] C. Deng, M. J. Er, G.-H. Yang, and N. Wang, ''Event-triggered consensus of linear multiagent systems with time-varying communication delays,'' *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 2916–2925, Jul. 2020.
[71] J. E. Normey-Rico and E. F. Camacho, *Control of Dead-Time Processes*. London, U.K.: Springer, 2007.
[72] A. Murilo, M. Alamir, and D. Alberer, ''A general NMPC framework for a diesel engine air path,'' *Int. J. Control*, vol. 87, no. 10, pp. 2194–2207, Apr. 2014.
[73] S. Sivananaithaperumal, ''PSO algorithm based nonlinear model predictive control,'' in *Proc. Int. Conf. Adv. Control Optim. Dyn. Syst. (ACODS)*, Feb. 2007, pp. 1–2.
[74] R. S. Rodrigues, ''Desenvolvimento de controlador preditivo para controle de atitude de satélites e validação em HIL,'' M.S. thesis, Dept. Engenharia Mecânica, Univ. de Brasília, Brasília, Brazil, 2018.
[75] M. Friedman, ''The use of ranks to avoid the assumption of normality implicit in the analysis of variance,'' *J. Amer. Stat. Assoc.*, vol. 32, no. 200, pp. 675–701, Dec. 1937.
[76] J. P. Shaffer, ''Modified sequentially rejective multiple test procedures,'' *J. Amer. Stat. Assoc.*, vol. 81, no. 395, pp. 826–831, Sep. 1986.
[77] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera, ''KEEL: A software tool to assess evolutionary algorithms for data mining problems,'' *Soft Comput.*, vol. 13, no. 3, pp. 307–318, Feb. 2009.
[78] I. Triguero, S. González, J. M. Moyano, S. García, J. Alcalá-Fdez, J. Luengo, A. Fernández, M. J. del Jesus, L. Sánchez, and F. Herrera, ''KEEL 3.0: An open source software for multi-stage analysis in data mining,'' *Int. J. Comput. Intell. Syst.*, vol. 10, pp. 1238–1249, Sep. 2017. [Online]. Available: http://www.keel.es

**EDUARDO DE MENDONÇA MESQUITA** was born in 1992. He received the B.S. degree in mechatronic engineering and the M.S. degree in mechatronic system from the Universidade de Brasília, in 2015 and 2018, respectively. He is currently pursuing the Ph.D. degree with the School of Electric and Computer Engineering, Universidade Federal de Goiás. His research interests include optimization problems and machine learning.

**RENATO CORAL SAMPAIO** (Member, IEEE) received the B.S. degree in mechatronic engineering, the M.S. degree in informatics, and the Ph.D. degree in mechatronic systems from the Universidade de Brasília, in 2004, 2013, and 2018, respectively. He is currently an Associate Professor with the Faculty of Gama (UnB Gama), Universidade de Brasília. His research interests include nonlinear predictive control, nonlinear dynamical systems, artificial neural networks, and support vector machine.


**HELON VICENTE HULTMANN AYALA** was born in 1986. He received the B.S. degree in control and automation engineering from the Pontifical Catholic University of Paraná (PUCPR), in 2009, the M.S. degree in advanced robotics from the Warsaw University of Technology and the University of Genoa, in 2012, and the Ph.D. degree in industrial and system engineering from PUCPR, in 2016. He has worked as an Undergraduate Intern with ZF Friedrichshafen AG, Germany, a Product Development Engineer with Embraer SA (Brazilian aircraft manufacturer), and a Research Staff Member with the IBM Research Brazil Laboratory. He joined the Department of Mechanical Engineering, Pontifical Catholic University of Rio de Janeiro, as a Professor Adjunto, in 2018. His research interests include system identification, advanced control, and machine learning.

**CARLOS HUMBERTO LLANOS** (Member, IEEE) received the B.S. degree in electrical engineering from the Universidad del Valle, Colombia, in 1983, the M.S. degree in computer science from the Universidade Federal de Minas Gerais, in 1990, and the Ph.D. degree in electrical engineering from the Universidade de São Paulo, in 1998. He is currently an Associate Professor with the Department of Mechanical Engineering, Universidade de Brasília, where he is also the Leader of the Automation and Control Group (GRACO). His research interests include reconfigurable systems, automation and control, automatic systems synthesis, instrumentation, building and home automation, and intelligent systems.