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

COB-2021-0055
HARRIS HAWKS OPTIMIZATION APPROACHES ON THE MULTIVARIABLE PID CONTROLLER TUNING

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

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

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

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

Abstract. *In this work, a performance comparison between the Harris Hawks Optimization (HHO) and four different created variants for the tuning of a decentralized proportional-integrative-derivative (PID) controller in a multiple-input multiple-output (MIMO) application is presented. The application has the objective to achieve optimal response on the control of a ball mill pulverizing system with a steady state decoupler, whose structure is composed of two inputs and two outputs. Metaheuristic approaches are suitable for this kind of problem due to its capacity to diversify the search space(exploration) and improve the quality of current solutions(exploitation). The HHO algorithm is a metaheuristic based on the cooperative hunting behavior of the Harris's Hawks, that consists in a tactic called "surprise pounce", in which each hawk attacks the prey from different positions and angles, cornering the target, exhausting it and finally diving to capture it when possible. Although the original HHO may present sufficient performance, it may be improved through addition of different techniques. To check for possible improvements, four HHO variants are implemented using different procedures, namely the use of cultural behavior based on normative and situational knowledges, oppositional-based learning on the second variant, covariance matrix learning on the third variant, and finally the application of quantum mechanics into the fourth one. The optimal parameter values for the PID controller are sought by minimizing the integral time squared error (ITSE) index of the response of the system. Simulations are performed using SIMULINK and MATLAB softwares. Statistical measures such as best, mean, median and standard deviation of the system response error for the tuned controllers are analyzed and compared over fifty runs. The obtained results have proven that the use of the previously mentioned proposals leads to improvements in the tuning efficiency of the HHO in this context, upgrading the performance of the PID controller for the control of the ball mill model.*
Keywords: *Proportional-Integrative-Derivative Control, Harris Hawks Optimization, Cultural Algorithm, Covariance Matrix Learning, Oppositional-Based Learning, Quantum Mechanics, Multiple-Input Multiple-Output Application*

1. INTRODUCTION

The proportional-integral-derivative (PID) is a simple, with clear functionality, and great applicability and efficiency controller, which has been widely used industrially since its invention in 1910 (Iruthayarajan and Baskar, 2009). PID popularity is due to its features such as the feedback loops, whose benefits include the capability of eliminating steady-state error adroitly, and the controller's ability to anticipate future samples because of its derivative action. Undoubtedly, PID controllers are suited to diverse control problems, mainly those whose process dynamics are uncertain or not completely acknowledged (Sabir and Ali, 2016).

The main objective of a PID controller is to reduce the error value between a required setpoint and the real measured process variable. Its operation is based on its input parameters, which are preset before the actual control activity (Fister et al., 2016). During the tuning process, the optimal values for these parameters are found, the most common approaches for this process are the Ziegler-Nichols and Cohen-Coon methods (Iruthayarajan and Baskar, 2009), although recently other techniques have become famous, such as neural networks (Hosseini et al., 2020), fuzzy-based approaches (Sain and Mohan, 2021), neuro-fuzzy (Shi et al., 2020) and evolutionary and swarm intelligence-based algorithms (Chang, 2007).

These last two approaches are metaheuristic optimization algorithms, methods capable of achieving a reasonable solution in a smaller computational time when compared to other approaches (Salcedo-Sanz, 2016). The Harris Hawks optimization (HHO) algorithm (Heidari et al., 2019) is a population-based metaheuristic, whose functionality mimics the cooperative hunting behavior of, as its name suggests, Harris's hawks. A large bird of prey commonly seen in the southwestern United States and Latin America. The algorithm simulates the search's exploration and exploitation by "attacking" the target from different angles and positions and it is known to have a strong exploitation capacity (Qu et al., 2020).

The contribution of this paper is to compare the performance of four HHO variants, such as the use of cultural behavior based on normative and situational knowledges, oppositional-based learning on the second variant, covariance matrix learning on the third variant, and application of quantum mechanics, in the tuning process of a PID controller for a ball mill pulverizing system with a steady-state decoupler. This application is defined as a multiple-input multiple-output (MIMO) process. The focus of the paper is on the possible improvements to the original algorithm with the application of the previously mentioned methods, not to find the universal best possible outputs of the system.

The remaining paper is organized as follows. Section 2 describes the fundamentals of a PID controller, its tuning process, as well as a description of the functionality of the HHO algorithm. In Section 3 the methodology used in this case study is emphasized. Sequentially, the results are discussed in Section 4. Finally, Section 5 embraces the final scopes and conclusion.

2. BRIEF REVIEW OF THEORETICAL FUNDAMENTALS

The standard PID controller is composed of proportional, integrative and derivative factors, which are combined to form the general controller structure as Eq. (1),

$$C(s)=K_P+\frac{K_I}{S}+K_D\cdot s, \tag{1}$$

where $\boldsymbol{K_P}$, $\boldsymbol{K_I}$ and $\boldsymbol{K_D}$ are the proportional, integrative and derivative gains, respectively. The first factor provides an action proportional to the error between the setpoint and the output, while the integrative one has the intent of decreasing the steady-state error of the system response and the last factor improves the transient response (Ogata, 2010).

For a multivariable process with $\boldsymbol{n}$ dimensions $\boldsymbol{G(s)}$, represented as an $\boldsymbol{n}$ x $\boldsymbol{n}$ matrix, it is necessary to have a same-size control matrix $\boldsymbol{K(s)}$, in which each element of the matrix $\boldsymbol{k_{ij}}$ is the control action of a controller, as defined previously in Eq. (1). However, as in this work the developed PID controller is decoupled, hence the control turns into a diagonal matrix as in Eq. (2).

$$K(s)=\begin{bmatrix} K_{11}(s) & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & K_{nn}(s) \end{bmatrix}, \tag{2}$$

2.1 Harris Hawks Optimization

Metaheuristics are a more comprehensive intuitive method for the solution of optimization problems (Rao, 2019). They may have different inspirations, like physics, human and nature behaviors, such as the case of evolutionary and swarm intelligence-based approaches. Metaheuristics are capable of diversifying the search space for solutions, in the exploration process, as well as improving current solutions by making slight modifications to their parameter values, in the exploitation process (Du and Swamy, 2016).

One such method is the HHO algorithm, which mimics the hunting behavior of the Harri's Hawks. These animals try to attack the prey from different directions, encircling the prey until it is exhausted and cannot escape, a tactic called "the seven kills" (Heidari et al., 2019). The exploration phase consists in the hawks searching for prey and the exploitation phase is when the hawks perform the “seven kills" tactic. The movement of the hawks also depends on the escaping energy of the prey. If the prey has enough energy to not get caught, the hawks perform a soft besiege, which means that they encircle the prey softly from all different positions, with the intent of tiring the prey. When the prey is tired, the hawks perform a hard besiege to finish their strategy, so they hardly encircle the prey to perform a surprise pounce, by making aggressive movements to get near the prey. The pseudocode of the HHO algorithm is shown in Figure 1.

Algorithm 1: Pseudocode for Harris Hawks Optimization (HHO)

Generate random initial population X_i with size N, and define number of generations T, upper bound UB, and lower bound LB
Evaluate initial solutions by calculating their fitness values
while stop criteria not reached **do**
  Assess fitness values of all members of the population
  for i = 1, 2, ...,N **do**
    Update initial energy $E_0 = 2\,rand(\,) - 1$
    Update jump strength $J = 2\,(1 - rand(\,))$
    Update prey energy $E = 2E_0(1 - \frac{t}{T})$
      if $|E| \geq 1$ **then** → *Exploration Phase*
        q = random number normally distributed between 0 and 1
        if q ≥ 0.5 **then**
          $X_i(t+1) = X_{rand}(t) - r_1 |X_{rand}(t) - 2r_2 X(t)|$
        else
          $X_i(t+1) = (X_{Rabbit}(t) - X_m(t)) - r_3(LB + r_4(UB - LB))$
        end if
      else → *Exploitation Phase*
        chance of the prey successfully escaping an attack r = random number normally distributed between 0 and 1
        if r ≥ 0.5 and $|E| \geq 0.5$ **then** → *Soft Besiege (SB)*
          $X_i(t+1) = (X_{Rabbit}(t) - X_i(t)) - E \cdot |J \cdot X_{Rabbit}(t) - X_i(t)|$
        else if r ≥ 0.5 and $|E| < 0.5$ **then** → *Hard Besiege (HB)*
          $X_i(t+1) = X_{Rabbit}(t) - E \cdot |X_i(t)|$
        else if r < 0.5 and $|E| \geq 0.5$ **then** → *SB with Rapid Dives*
          $X_i(t+1) = Y = X_{Rabbit}(t) - E \cdot |J \cdot X_{Rabbit}(t) - X_i(t)|$ **if** *Fitness (Y) < Fitness* $(X_i(t))$
          $X_i(t+1) = Z = X_{Rabbit}(t) - E \cdot |J \cdot X_{Rabbit}(t) - X_i(t)| + S \times LF\,(D)$ **if** *Fitness (Z) < Fitness* $(X_i(t))$
        else if r < 0.5 and $|E| < 0.5$ **then** → *HB with Rapid Dives*
          $X_i(t+1) = Y = X_{Rabbit}(t) - E \cdot |J \cdot X_{Rabbit}(t) - X_m(t)|$ **if** *Fitness (Y) < Fitness* $(X_i(t))$
          $X_i(t+1) = Z = X_{Rabbit}(t) - E \cdot |J \cdot X_{Rabbit}(t) - X_m(t)| + S \times LF\,(D)$ **if** *Fitness (Z) < Fitness* $(X_i(t))$
        end if
      end if
  end for
  Return X_{Rabbit}
end while

Figure 1. Pseudocode of the HHO algorithm.

As can be seen in the Figure above, after defining the parameters of population and number of generations, the algorithm begins by randomly generating and evaluating the initial population. The general loop consists in initially recording the best solution as $\boldsymbol{X_{Rabbit}}$. Each hawk receives a value $\boldsymbol{E}$ to model the escaping energy of the prey, which also depends on the current generation $\boldsymbol{t}$. Next, the solution position $\boldsymbol{X_i}$ is updated according to $\boldsymbol{E}$. If the prey's energy is greater than or equal to one, the hawk is in the exploration phase, so its position is updated, this value also depends on an $\boldsymbol{X_{rand}}$ variable, representing a randomly selected hawk in the population, and the variable $\boldsymbol{X_m}$, which consists of the mean position of the hawk population. Some random constants normally distributed between zero and one, $\boldsymbol{r_1}$, $\boldsymbol{r_2}$, $\boldsymbol{r_3}$, $\boldsymbol{r_4}$ and $\boldsymbol{q}$ are also generated.

The exploitation phase happens when the prey's energy is lower than one, meaning that the prey has lost most of its energy and that it is now susceptible to an attack. In this phase, a soft or a hard besiege may be performed by the hawk depending on the prey's remaining energy. When $|\boldsymbol{E}|$ is greater than or equal to 0.5, a soft besiege happens with the intent of tiring the prey until the surprise pounce occurs, following the same idea, when $|\boldsymbol{E}|$ is lower than 0.5 the hawk performs a hard besiege to quickly finish the prey. However, the hawks are not the only agents moving. As we know, the prey is also trying to escape from the reach of the hawks. To model the chance of the prey successfully escaping an attack, a random number $\boldsymbol{r}$ also normally distributed between zero and one is used. When $|\boldsymbol{E}|$ is greater than or equal to 0.5 and $\boldsymbol{r}$ is greater than 0.5, the prey still has some energy left and tries to escape but fails to do so. In this case, the hawks perform a soft besiege. When $|\boldsymbol{E}|$ is lower than 0.5 and $\boldsymbol{r}$ is greater than or equal to 0.5 the prey is too exhausted to be able to escape, then a hard besiege is done by the hawks so that they can perform the surprise pounce. When $\boldsymbol{r}$ is lower than 0.5, the rabbit is still able to escape.

To model its varying, flexible movements, as well as the rapid dives the hawks make to adapt to the motion of the prey, the levy flight distribution is used in the algorithm. So, when $|\boldsymbol{E}|$ is greater than or equal to 0.5, the hawk performs a soft besiege with progressive rapid dives. Firstly, calculating the decision-making hawks perform, supposing they can take the best possible move. They take this movement only if the resulting fitness evaluation is better than their previous position. If false, they start diving asymmetrically, mathematically modeled by the levy flight function, taken only if the resulting fitness evaluation is better than the current position. In this case, $\boldsymbol{S}$ is a random vector with size 1 x $\boldsymbol{D}$ (dimension of the problem), and $\boldsymbol{LF}$ is the levy flight function.

In the case when r is lower than 0.5 but $|\boldsymbol{E}|$ is lower than 0.5, the prey does not have enough energy to escape, and the hawks perform rapid dives in a hard besiege, to aggressively pursue and approximate the prey. The next position is then calculated following the same logic of the previous case.

2.2 Cultural Harris Hawks Optimization Algorithm

Cultural algorithms make use of domain knowledge to remove generality in the problem and accelerate convergence, improving the performance of an evolutionary algorithm (Becerra and Coello, 2006). In a standard evolutionary algorithm, there is a population space that contains the possible solutions for the problem. In a cultural evolutionary algorithm, there is one space more: the belief space. It contains information taken from any solution in the population, allowing other solutions to use this information in the evolution process. The belief space is composed of knowledge sources. These sources may be situational, normative, topographical, or historical. In the Cultural Harris Hawks Optimization (CHHO) two knowledge sources are implemented: the situational knowledge, which consists of the best solution, and the normative knowledge, which is a set of intervals for decision variables. As it can be seen in Figure 1, the situational knowledge is already used in the original HHO, so the CHHO adds the normative knowledge.

The normative knowledge controls ***LB*** and ***UB*** (vectors with variable lower and upper bounds in each position), ***L*** and ***U*** (vectors with fitness function values for ***LB*** and ***UB*** respectively), and dm, a scaling factor to change the mutation operator in the exploitation phase. The CHHO is shown in Figure 2.

Algorithm 1: Pseudocode for Cultural Harris Hawks Optimization (CHHO)

```
Generate random initial population X_i with size N, where each X_i has NV variables, and define number of generations T, upper bound UB, and lower bound LB
Evaluate initial solutions by calculating their fitness values
while stop criteria not reached do
    Assess fitness values of all members of the population
    for i = 1, 2, ..., N do
        Update initial energy E_0 = 2 rand( ) − 1
        Update jump strength J = 2 (1 − rand( ))
        Update prey energy E = 2E_0(1 − t/T)
            if |E| ≥ 1 then ──────────────────────────────────→ Exploration Phase
                q = random number normally distributed between 0 and 1
                if q ≥ 0.5 then
                    X_i(t + 1) = X_rand(t) − r_1 | X_rand(t) − 2r_2 X(t) |
                else
                    X_i(t + 1) = (X_Rabbit(t) − X_m(t)) − r_3(LB + r_4(UB − LB))
                end if
            else ─────────────────────────────────────────────→ Exploitation Phase
                chance of the prey successfully escaping an attack r = random number normally distributed between 0 and 1
                if r ≥ 0.5 and |E| ≥ 0.5 then ────────────────→ Soft Besiege (SB)
                    for v = 1, 2, NV then
                        if X_i,v(t) < LB_v(t) then
                            X_i,v(t + 1) = (X_Rabbit,v(t) − X_i,v(t)) + E| J X_Rabbit,v(t) − Xi_,v(t)|
                        else if X_i,v(t) > UB_v(t) then
                            X_i,v(t + 1) = (X_Rabbit,v(t) − X_i,v(t)) − E| J X_Rabbit,v(t) − Xi_,v(t)|
                        else
                            X_i,v(t + 1) = (X_Rabbit,v(t) − X_i,v(t)) + ((UB_v(t) − LB_v(t))/dm_v(t))E(J X_Rabbit,v(t) − X_i,v(t))
                        end if
                    end for
                else if r >= 0.5 and |E| < 0.5 then ──────────→ Hard Besiege (HB)
                    for v = 1, 2, ..., NV
                        if X_i,v(t) < LB_v(t)
                            X_i,v(t + 1) = X_Rabbit,v(t) + E| X_Rabbit,v(t) − Xi_,v(t)|
                        else if X_i,v(t) > UB_v(t)
                            X_i,v(t + 1) = X_Rabbit,v(t) − E| X_Rabbit,v(t) − X_i,v(t)|
                        else
                            X_i,v(t + 1) = X_Rabbit,v(t) + ((UB_v(t) − LB_v(t))/dm_v(t))E(X_Rabbit,v(t) − X_i,v(t))
                        end if
                    end for
                else if r < 0.5 and |E| >= 0.5 ───────────────→ SB with Rapid Dives
                    for v = 1, 2, ..., NV
                        if X_i,v(t) < LB_v(t)
                            Y_v(t) = X_Rabbit,v(t) + E| J X_Rabbit,v(t) − X_i,v(t)|
                        else if X_i,v(t) > UB_v(t)
                            Y_v(t) = X_Rabbit,v(t) − E| J X_Rabbit,v(t) − X_i,v(t)|
                        else
                            Y_v(t) = X_Rabbit,v(t) − ((UB_v(t) − LB_v(t))/dm_v(t))E| J X_Rabbit,v(t) − X_i,v(t)|
                        end if
                    end for
                    if Fitness(Y) < Fitness(Xi(t))
                        X_i(t + 1) = Y
                    else
                        S = rand(1, nV)
                        Z = Y + S.* levy(NV)
                        if Fitness(Z) < Fitness(Xi(t))
                            X_i(t + 1) = Z
                            Fitness(Xi(t + 1)) = Fitness(Z)
                        end if
                    end if
                elseif r < 0.5 and |E| < 0.5 ─────────────────→ HB with Rapid Dives
                    for v = 1, 2, ..., NV
                        if X_i,v(t) < LB_v(t)
                            Y_v(t) = X_Rabbit,v(t) + E| J X_Rabbit,v(t) − X_m(t)|
                        else if X_i,v(t) > UB_v(t)
                            Y_v(t) = X_Rabbit,v(t) − E| J X_Rabbit,v(t) − X_m(t)|
                        else
                            Y_v(t) = X_Rabbit,v(t) − ((UB_v(t) − LB_v(t))/dm_v(t))E(J X_Rabbit,v(t) − X_m(t))
                        end if
                    end for
                    if Fitness(Y) < Fitness(Xi(t))
                        X_i(t + 1) = Y
                    else
                        S = rand(1, nV)
                        Z = Y + S.* levy(NV)
                        if Fitness(Z) < Fitness(Xi(t))
                            X_i(t + 1) = Z
                            Fitness(Xi(t + 1)) = Fitness(Z)
                        end if
                    end if
                end if
    end for
    Update LB, UB, L and U
end while
Return X_Rabbit
```

Figure 2. Pseudocode of the CHHO algorithm.

An important part of the normative knowledge is the update of ***LB***, ***UB***, ***L*** and ***U*** at the end of every iteration in the main 'while' loop.

2.3 Covariance Matrix Harris Hawks Optimization Algorithm

The covariance matrix reflects the diversity of the population and the interactions between variables. This relation can be used to attenuate the dependency of the algorithm on the coordinate system and to loosen interactions among the variables (Wang et al., 2014). The process has two main steps: decomposing the covariance matrix in its eigenvalue components and transforming the vector coordinates. Covariance Matrix Harris Hawks Optimization (CMHHO) uses this concept in the exploration phase, as shown in Figure 3.

Algorithm 3: Pseudocode for Covariance Matrix Harris Hawks Optimization (CMHHO)

```
Generate random initial population X_i with size N, and define number of generations T, upper bound UB, and lower bound LB
Evaluate initial solutions by calculating their fitness values
while stop criteria not reached do
    Assess fitness values of all members of the population
    for i = 1, 2, ...,N do
        Update initial energy E_0 = 2 rand( ) − 1
        Update jump strength J = 2 (1 − rand( ))
        Update prey energy E = 2E_0(1 − t/T)
        Calculate covariance matrix C from the top half best solutions
        B = eigenvalues(C)
        if |E| ≥ 1 then  ————→ Exploration Phase
            q = random number normally distributed between 0 and 1
            if q ≥ 0.5 then
                V_i(t) = X_rand(t) − r_1| X_rand(t) − 2r_2X(t) |
            else
                V_i(t = (X_Rabbit(t) − X_m(t)) − r_1|LB + r_2(UB − LB)|
            end if
            X'_i(t) = B^−1 · X_i(t)
            V'_i(t) = B^−1 · V_i(t)
            r = random number normally distributed between 0 and 1
            i = random number normally distributed between 0 and N
            if r ≥ 0.5 or i = i_rand then
                U'_i(t) = V'_i(t)
            else
                U'_i(t) = X'_i(t)
            end if
            U_i(t) = B · U'_i(t)
            X_i(t + 1) = U_i(t) if Fitness (U_i(t)) < Fitness (X_i(t))
        else  ————→ Exploitation Phase
            chance of the prey successfully escaping an attack r = random number normally distributed between 0 and 1
            if r ≥ 0.5 and |E| ≥0.5 then  ————→ Soft Besiege (SB)
                X_i(t + 1) = (X_Rabbit(t) − X_i(t)) − E · |J · X_Rabbit(t) − X_i(t)|
            else if r ≥ 0.5 and |E| < 0.5 then  ————→ Hard Besiege (HB)
                X_i(t + 1) = X_Rabbit(t) − E · |X_i(t)|
            else if r < 0.5 and |E| ≥ 0.5 then  ————→ SB with Rapid Dives
                X_i(t + 1) = Y = X_Rabbit(t) − E · |J · X_Rabbit(t) − X_i(t)| if Fitness (Y) < Fitness (X_i(t))
                X_i(t + 1) = Z = X_Rabbit(t) − E · |J · X_Rabbit(t) − X_i(t)| + S × LF (D) if Fitness (Z) < Fitness (X_i(t))
            else if r < 0.5 and |E| < 0.5 then  ————→ HB with Rapid Dives
                X_i(t + 1) = Y = X_Rabbit(t) − E · |J · X_Rabbit(t) − X_m(t)| if Fitness (Y) < Fitness (X_i(t))
                X_i(t + 1) = Z = X_Rabbit(t) − E · |J · X_Rabbit(t) − X_m(t)| + S × LF (D) if Fitness (Z) < Fitness (X_i(t))
            end if
        end if
    end for
    Return X_Rabbit
end while
```

Figure 3. Pseudocode of the CMHHO algorithm.

2.4 Quantum Matrix Harris Hawks Optimization Algorithm

Quantum Harris Hawks Optimization (QHHO) is based on quantum mechanics. More specifically, it makes use of the same strategy present in the Quantum Particle Swarm Optimization (QPSO), which means it is based on a quantum delta potential well model, as well as the mean best solution in the population (Sun et al., 2016).

This quantum strategy is implemented in the exploration phase of the HHO. When q is greater than or equal to 0.5, the equation changes to Eq (3),

$$X_i(t+1) = X_{Rabbit} + r_1 \left| X_{Rabbit} - X_i(t) \right| \cdot \log\left(\frac{1}{r_2}\right), \tag{3}$$

and when q is lower than 0.5, it changes to Eq. (4),

$$X_i(t+1)=X_{Rabbit}-r_1\left|X_{Rabbit}-X_i(t)\right|\cdot\log\left(\frac{1}{r_2}\right), \tag{4}$$

where $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are random numbers in (0,1). This strategy was implemented only to improve the search in the exploration phase, so the rest of the algorithm remains the same.

2.5 Oppositional Matrix Harris Hawks Optimization Algorithm

According to probability theory, 50% of the time a guess is further from the solution than its opposite guess (Xu et al., 2014). Every time a new solution is generated, oppositional-based learning generates its opposite and takes only the best one. Mathematically, for a solution vector $\boldsymbol{X}$, the opposite solution would be calculated by the opposite solution $\boldsymbol{OX}$, as noted in Eq (5).

$$OX=LB+UB-X, \tag{5}$$

The Oppositional Harris Hawks Optimization (OHHO) generates the opposite of every newly generated vector $\boldsymbol{X}$ in the algorithm in Figure 1, including the randomly generated initial solutions. It compares the two solutions and takes the one with the lower fitness function value as the new solution. The rest of the process is the same.

2.6 Case Study: Ball Mill Pulverizing System

The ball mill pulverizing system model is described as Eq. (6), therefore not a diagonally row dominant matrix.

$$G(s)=\begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix}=\begin{bmatrix} \frac{3.5}{(80\cdot s+1)^3} & \frac{-0,14}{(60\cdot s+1)^2} \\ \frac{-2}{(8\cdot s+1)^2} & \frac{-18}{(10\cdot s+1)} \end{bmatrix}, \tag{6}$$

In order to reduce the impact of interaction among processes, a steady-state decoupler is given as shown in Eq. (7)

$$D(s)=G^{-1}(0)=\begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}=\begin{bmatrix} 0.1378 & -0.1538 \\ -2.1978 & -3.8462 \end{bmatrix}, \tag{7}$$

As it can be seen, the process is a multiple inputs, multiple outputs (MIMO) system with two inputs and two outputs, making the PID structure as seen in Eq. (8), with a control process according to Figure 4.

$$K(s)=\begin{bmatrix} K_{P1}+\frac{K_{I1}}{S}+K_{D1}\cdot s & 0 \\ 0 & K_{P2}+\frac{K_{I2}}{S}+K_{D2}\cdot s \end{bmatrix}, \tag{8}$$

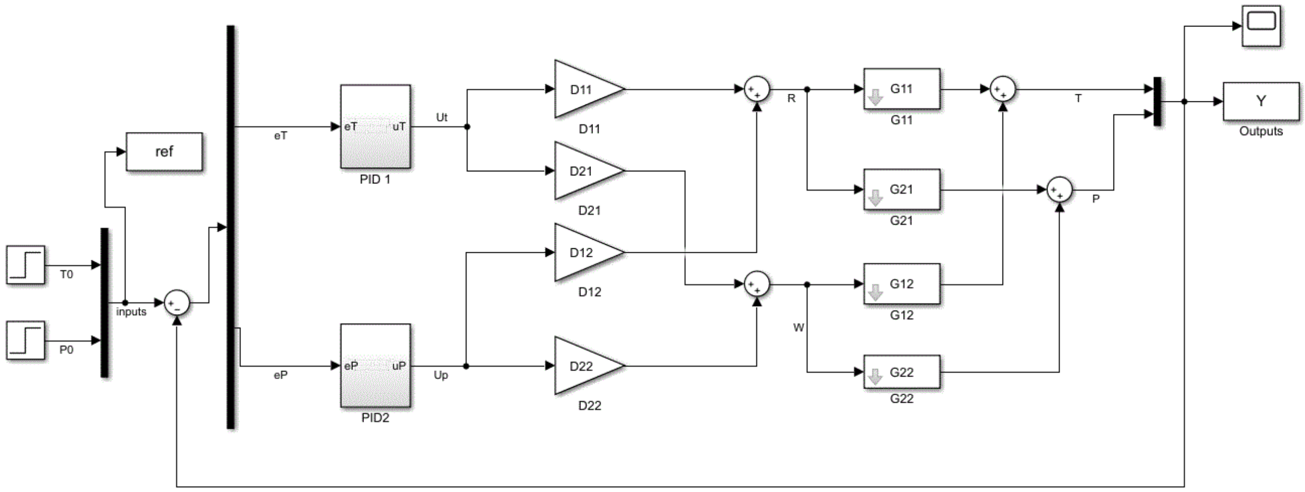

Figure 4. Structure of the control procedure.

3. METHODOLOGY

To obtain the best PID parameters for the control of the ball mill pulverizing system, the solution vector (chromosome) representation in the metaheuristic optimization algorithms is defined in Figure 5. The first three elements represent the values for the first PID controller, and the next three are the elements for the second PID controller.

K_{P1}	K_{I1}	K_{D1}	K_{P2}	K_{I2}	K_{D2}

Figure 5. Solution vector in the optimization algorithm.

The lower and upper bounds of each of the six parameters are 0 and five, respectively. The algorithm limits the parameter values as the process goes along. The fitness function used was the sum of the integral time squared error (ITSE) indexes for each output, which is defined in Eq. (9).

$$F(x)=\sum_{i=1}^{2}\left(\int_{t=0}^{T} t\cdot e_i^2(t)\,dt\right), \tag{9}$$

where $\boldsymbol{X}$ is the current solution, $\boldsymbol{t}$ and $\boldsymbol{T}$ are the current and maximum time (in this case 100 seconds) respectively, and $\boldsymbol{e}_i$ is the error for output number $\boldsymbol{i}$.

In this work, simulations were run in the MATLAB-SIMULINK environment. The hardware is composed of an Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz, 8GB RAM PC. The simulations are run with a population size of 50, over 100 generations, and a sample time of 0.1 second. To get better comparisons between the results for each algorithm, each algorithm is run 50 times, and mean, median, minimum, maximum and standard deviation values for the fitness function are computed.

4. RESULTS AND DISCUSSION

The statistical results of the fitness function values for each HHO variation are present in Table 1. The minimum, maximum, mean, median and standard deviation values over 50 runs serve for comparison between the different techniques. Although the ITSE values of the results are very close, every new variation had better results than the original HHO, with the OHHO being the best one. So the main objective of the paper, which was to verify the improvements of the variants, was accomplished. The range of values in the OHHO was smaller, as it can be seen by the minimum and maximum values, as well as the standard deviation. The CMHHO has the best mean and median values, but it could not be more efficient than the OHHO. The responses for outputs 1 and 2 are in Figures 4 and 5.

Table 1. Experimental results for the simulations of the HHO variations.

Algorithm	Minimum	Maximum	Mean	Median	Standard deviation
HHO	2542.224	2764.807	2603.770	2597.031	50.576
CMHHO	2541.107	2711.463	2577.742	2554.090	47.270
CHHO	2540.484	22381.640	2984.077	2559.720	2799.666
QHHO	2540.400	2737.795	2605.887	2608.774	54.465
OHHO	2540.372	2708.003	2596.122	2580.987	45.863

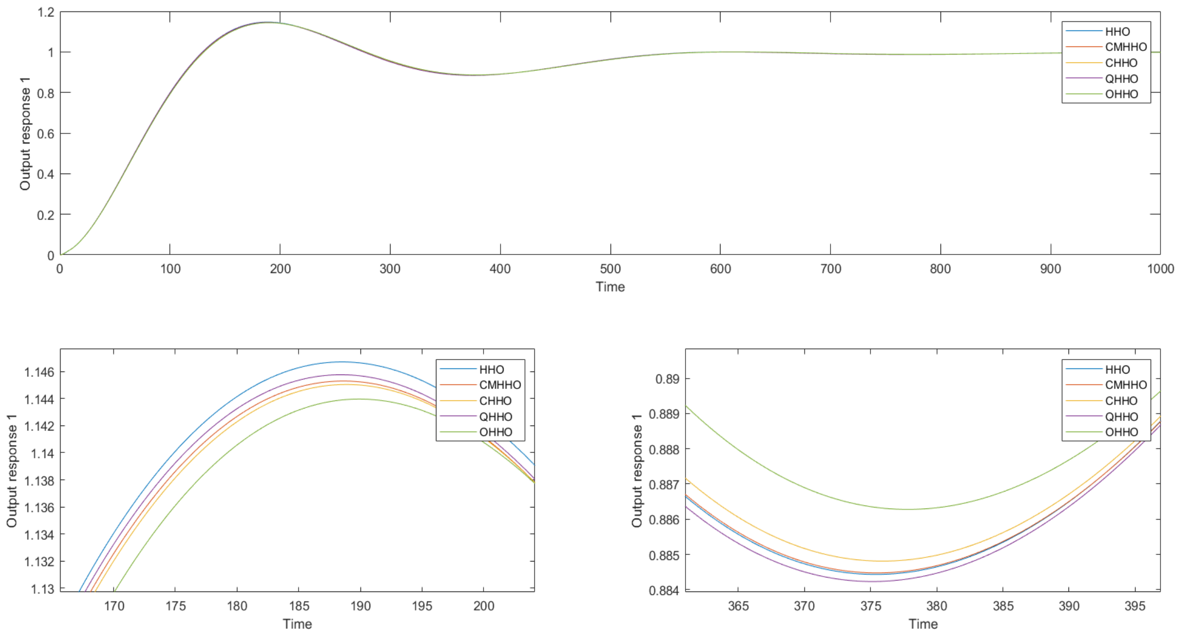

Figure 4. Responses for output 1.

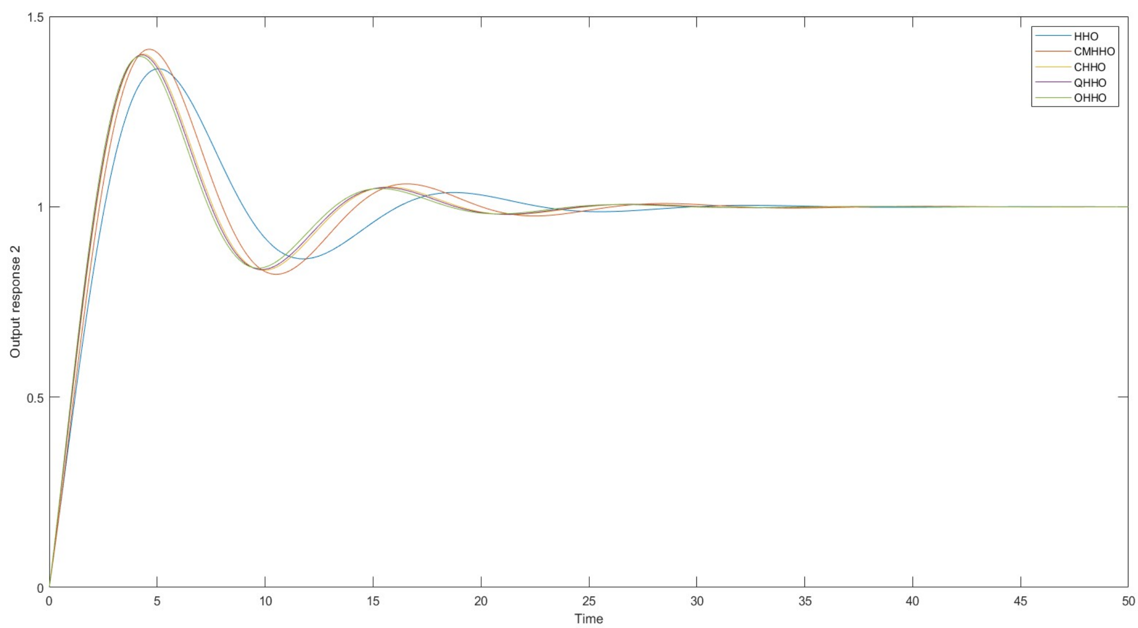

Figure 5. Responses for output 2.

5. FINAL CONSIDERATIONS

Each metaheuristic algorithm is influenced by the problem to be solved and its available information. Even though the HHO algorithm is a relatively new method when compared to other algorithms, it has proven to be capable of tuning a decentralized PID control for a MIMO plant effectively, as well as its four created variants. The created variants all showed improvement when compared to the original algorithm, and the OHHO was the most efficient one. Further research may focus on improving the efficiency of the algorithms, as well as developing new variants.

6. ACKNOWLEDGEMENTS

This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil* (CAPES) – Finance Code 001.

7. REFERENCES

Becerra, R.L. and Coello, C.A.C., 2006. “Cultured differential evolution for constrained optimization”. Computer *Methods in Applied Mechanics and Engineering*, Vol. 195, No. 33 – 36, pp. 4303 – 4322.

Chang, W. D., 2007. “A multi-crossover genetic approach to multivariable PID controllers tuning”. *Expert Systems with Applications*, Vol. 33, pp. 620 – 626.

Du, K. L., Swamy, M. N. S., 2016. “Search and Optimization by Metaheuristics”. Birkhäuser: Montreal, Canada.

Fister, D., Fister Jr, I., Fister, I., Šafarič, R., 2016. “Parameter tuning of PID controller with reactive nature-inspired algorithms”. *Robotics and Autonomous Systems*, Vol. 84, pp. 64 – 75.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. “Harris hawks optimization: Algorithm and applications”. *Future generation computer systems*, Vol. 97, pp. 849 – 872.

Hosseini, S. A., Shirani, A. S., Lotfi, M., Menhaj, M. B., 2020. “Design and application of supervisory control based on neural network PID controllers for pressurizer system”. *Progress in Nuclear Energy*, Vol. 130, pp. 103570.

Iruthayarajan, M. W., Baskar, S., 2009. “Evolutionary algorithms based design of multivariable PID controller”. *Expert Systems with applications*, Vol. 36, No. 5, pp. 9159 – 9167.

Ogata, K., 2010. “Modern control engineering”. Prentice hall.

Rao, S.S., 2019. “Engineering optimization: theory and practice”. John Wiley & Sons.

Sabir, M. M., Ali, T., 2016. “Optimal PID controller design through swarm intelligence algorithms for sun tracking system”. *Applied Mathematics and Computation*, Vol. 274, pp. 690 – 699.

Sain, D., Mohan, B. M., 2021. “A simple approach to mathematical modelling of integer order and fractional order fuzzy PID controllers using one-dimensional input space and their experimental realization”. *Journal of the Franklin Institute*, In Press.

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

Shi, Q., Lam, H. K., Xuan, C., Chen, M., 2020. “Adaptive neuro-fuzzy PID controller based on twin delayed deep deterministic policy gradient algorithm”. *Neurocomputing*, Vol. 402, pp. 183 – 194.

Sun, J., Lai, C.H. and Wu, X.J., 2016. “Particle swarm optimisation: classical and quantum perspectives”. *Crc Press*.

Qu, C., He, W., Peng, X., Peng, X., 2020. “Harris hawks optimization with information exchange”. *Applied Mathematical Modelling*, Vol. 84, pp. 52 – 75.

Wang, Y., Li, H.X., Huang, T. and Li, L., 2014. “Differential evolution based on covariance matrix learning and bimodal distribution parameter setting”. *Applied Soft Computing*, Vol. 18, pp. 232 – 247.

Xu, Q., Wang, L., Wang, N., Hei, X. and Zhao, L., 2014. “A review of opposition-based learning from 2005 to 2012”. *Engineering Applications of Artificial Intelligence*, Vol. 29, pp. 1 – 12.

8. RESPONSIBILITY NOTICE

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