IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 September 2022, accepted 8 October 2022, date of publication 14 October 2022, date of current version 21 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3214850

APPLIED RESEARCH

Kinematic Optimization of 6DOF Serial Robot Arms by Bio-Inspired Algorithms

ERVIN GALAN-URIBE AND LUIS MORALES-VELAZQUEZ, (Member, IEEE)
Facultad de Ingeniería, Universidad Autónoma de Querétaro, Campus San Juan del Río, San Juan del Río, Santiago de Querétaro 76807, Mexico

Corresponding author: Luis Morales-Velazquez (luis.moralesv@uaq.mx)

This work was supported by the Consejo Nacional de Ciencia y Tecnología (CONACYT) under Grant 783320 and Grant FONDEC-UAQ-2021-LMV-6829.

ABSTRACT Robotic systems are essential to technological development in the industrial, medical, and aerospace sectors. Nevertheless, their use in different applications requires that the robot have the best possible execution efficiency. For this, a robot with specific optimized characteristics is necessary to impact the performance of the specific task. Different techniques have been used in robot optimization, the most widely used being genetic algorithms (GA) and particle swarm optimization (PSO). However, there are optimization algorithms with high convergence speeds inspired by animal behavior whose application in robot optimization has not been reported. In this work, bio-inspired algorithms Harris hawks optimization (HHO) and Grey wolf optimizer (GWO) are applied to a six-degree-of-freedom (6DOF) robot arm design through kinematic optimization. The lengths of the main robot links are optimized to improve the workspace volume and obtain a better-conditioned robot using the structural length index (SLI) and global condition index (GCI) as objective functions. A comparison is made between the proposed algorithms and GA and PSO regarding convergence speed, computational load, and optimality. Similar behavior has been found for HHO, GWO, and PSO compared to GA for both indexes. For the GCI problem, an average improvement of 14% was found when optimizing an industrial robot arm. Furthermore, multiple runs experiment is performed to test the robustness of the algorithms. The results show that HHO is the best technique to obtain an optimal robot design because it needs less than ten iterations to provide a better result despite its computational load.

INDEX TERMS Evolutionary computation, genetic algorithms, grey wolf optimizer, Harris hawk optimizer, particle swarm optimization, robot kinematics.

I. INTRODUCTION

Robotic systems can be described as the combination of three central systems: a mechanical, an electronic, and a computational system working together to perform the required tasks [1]. However, the mechanical system interacts directly with the environment and is subject to physical restrictions that determine the design of the robot. These limitations are resolved by optimizing robot performance indices based on physical characteristics, such as link length, workspace, degrees of freedom, mass, stiffness, and inertia. Optimization methodologies based on bio-inspired techniques are widely used to solve robot mechanical design problems.

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

Different robot design methodologies have been proposed using the combination of optimization algorithms and performance indices as objective functions. Some of them report algorithms such as MultiStart to optimize manipulability index [2], minimization of dexterity index [3], and exhaustive method to optimize volume, workspace, dexterity, and static efficiency [4]. Robot design proposals based on bio-inspired algorithms like particle swarm optimization (PSO) have been reported in the literature. For example, optimizing the mass and stiffness of a five-degree-of-freedom (5DOF) parallel kinematic machine obtains a significant improvement in both parameters [5]. Also, the optimization of the kinematic structures of a hybrid parallel robot using the dexterity index to obtain a better singularity-free workspace compared to the general Stewart platform is presented in [6]. In [7] is

presented the optimization of the installation position of the hydraulic cylinder for wearable medical robotics meeting the requirements of normal human movement. Also, [8] presents the optimal design parameters for a lower-limb assistance device for walking; minimizing the load on human articulations. Another example is [9], which optimizes the mechanical parameters of a wearable robotic limb for hemiplegia rehabilitation, meeting the requirement to be used for humans. Finally, in [10], the design variables of a dual cable-suspended robot for construction are optimized to avoid collisions and obtain a sizeable free-collision workspace.

Recent work on optimal robot design has also reported the use of genetic algorithms (GA) to optimize different parameters, for example, the manipulator parameters of 7DOF serial robot using structural length index (SLI), global conditioning Index (GCI), and Modified Dynamic Conditioning Index (MDCI) improving the kinematic and dynamic robot performance [11]. In [12] is presented the optimization of the structural design of a parallel Delta robot using the Stiffness Evaluation Index (SEI), improving the robot dynamics. Also, in [13], the geometry and workspace of a tree-type robotic system were optimized with GA, and the approach is checked with the design of the robotic platform and a grasping system. The optimal design of a pneumatic robot considering the dexterity and load capacity is made in [14]. In [15], radius optimization between joints of an 8DOF upper-limb exoskeleton is made to improve the minimum volume and maximum dexterity in the workspace using GA. The optimal design of an under-actuated robotic gripper improving the dimensions of the links and location of pulleys to achieve a stable grasp performance is presented in [16]. Another example is the optimization of the limited workspace of a Stewart robotic platform using GA and neural networks to obtain the maximum workspace possible and the forward kinematics model, respectively, presented in [17]. Taking into account both the stiffness of irregular-shaped inflatable tubes in [18] is performed the optimal design of a joint for inflatable robotic arms focused on surveying disaster environments. Finally, the optimal design of medical pair of instruments for endoscopic operations using recorded suturing, anatomical data, and the distance between the robotic tip and the desired position is presented in [19].

Through the works presented, it is observed that the design of robots continues to be carried out despite being an area exploited for many years. In addition, most of the works that use optimization algorithms for the problem of robot design report the use of genetic algorithms and particle swarm optimization, both the best-known and most used optimization methods. Due to their relevance, they have been extensively studied and modified to improve their performance. One clear example of PSO is [20], where it is proposed that the particles select their respective best leader through a hybrid scheme, thus improving the convergence and diversity of a multi-objective PSO. A review of GA and PSO algorithms, their variants and applications can be found in [21] and [22] respectively. On the other hand, the application of different recent bio-inspired algorithms in the specific problem of robot design and optimization, besides their advantages and disadvantages, has not been reported in the literature.

Currently, there are different bio-inspired optimization techniques widely used in other areas of science, like Grey wolf optimizer (GWO) and Harris hawks optimization algorithm (HHO). These algorithms have advantages such as fast convergence, reduced computational load, and ease of implementation. The GWO algorithm is based on a wolves pack behavior [23] and has been used in the robotics area in distinct applications as robot trajectory planning [24], [25], [26], collision avoidance in mobile robots [27], robot control [28] and inverse kinematics calculation [29]. The HHO is based on the hunting behavior of the Harris hawk [30] and has a high performance in optimization problems. Recent applications of HHO are: vehicle design [31], classification and feature optimization [32], optimization of a long short-term memory neural network [33], optimization of scheduling problems focused on energy consumption [34], and airplane control [35]. The works presented show some of the recent applications of the GWO and HHO algorithms and their performance in different optimization problems. However, few works report computational time or hardware where the implementations are carried out. This information helps decide which technique to use in the design process. Despite the advantages of these algorithms, their application has not been reported in the robot manipulator design and optimization problem.

The main contribution of this work is to incorporate the GWO and HHO bio-inspired algorithms in the optimal design of robot manipulators since their advantages have not been reported in the literature. In addition, a comparison with frequently used PSO and GA algorithms helps to demonstrate its benefits. To this end, two cases of kinematic optimization of a 6DOF serial robotic arm are proposed through two performance indices that will be used as objective functions to be optimized. Considering the convergence speed and computational load of the different algorithms, the tests show that the optimization of the robot through proposed bio-inspired algorithms has better results and faster convergence than GA and PSO.

II. THEORETICAL BACKGROUND

The 6DOF serial robot arm (Fig. 1a) is one of the most used in industry and academia, and its kinematic analysis is present in most robotics books. Below is a brief review of robot performance indexes used as objective functions in the optimization process. A detailed analysis of robot kinematics concepts and their properties can be found in [36]. In addition, a brief review of bio-inspired algorithms used in robot kinematic optimization is presented.

A. ROBOT PARAMETERS

For this work, a simplified 6DOF serial robot arm is proposed for the optimization problem. The standard Denavit–Hartenberg notation (DH) is used, a graphical

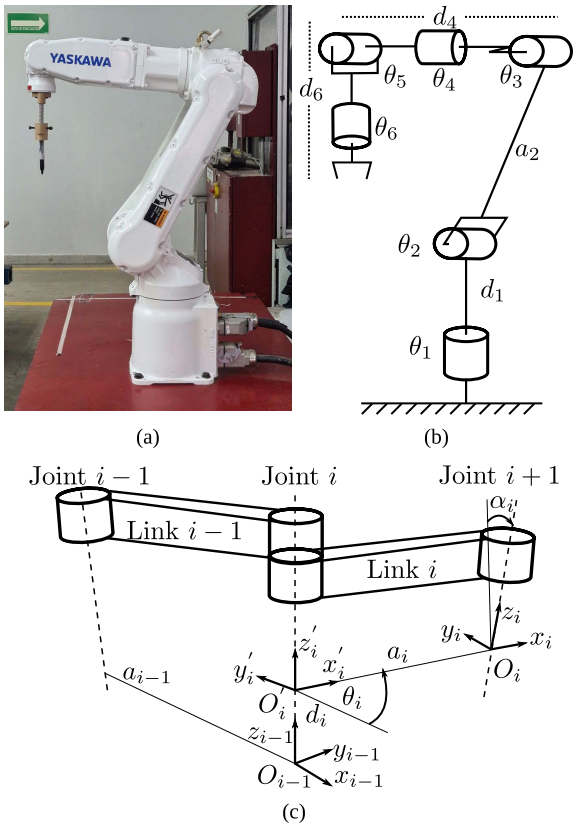

FIGURE 1. (a) 6DOF Yaskawa Motoman-MH5LS II robot arm, (b) wire diagram representation of a simplified 6DOF robot, (c) graphical representation of DH kinematic parameters.

TABLE 1. Standard denavint-hatenberg parameters for a standard 6DOF robot arm.

Joint	θ_i	d_i	a_i	α_i
1	θ_1	d_1	0	$\pi/2$
2	θ_2	0	a_2	0
3	θ_3	0	0	$\pi/2$
4	θ_4	d_4	0	$-\pi/2$
5	θ_5	0	0	$\pi/2$
6	θ_6	d_6	0	0

representation for the robot arm is shown in Fig. 1(b), and the DH parameters are shown in Table 1.

According to DH notation, for the Table 1 values:

- θ_i is the angle between axes x_{i-1} and x_i about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.
- d_i is the coordinate of O_i' along z_{i-1}.
- a_i is the distance between O_i and O_i'.
- α_i is the angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise.

With the parameters of Table 1, the computation of forward kinematics is obtained through the homogeneous transformation matrices. The Jacobian matrix is obtained from forward kinematics with the method presented in [36].

B. STRUCTURAL LENGTH INDEX

The Structural Length Index (SLI) is an index that presents the relationship between the length of the robot links and the volume of the workspace W. It is defined as (1):

$$SLI = \frac{L}{\sqrt[3]{W}}. \tag{1}$$

From (1) L is the length sum of the links a_{i-1} and offsets d_{i-1} (2).

$$L = \sum_{i=1}^{n} a_{i-1} + d_{i-1} \tag{2}$$

A desirable design has a low SLI through a large workspace volume and a small link length.

C. GLOBAL CONDITION INDEX

The Global Condition Index (GCI) proposed by Gosselin and Angeles in [37] is a workspace adaptation of the condition number index introduced by Salisbury and Craig in [38]. The condition number measures the independence between the columns of the Jacobian matrix [39]. For a full-rank Jacobian matrix, the condition number is defined by the ratio of the Jacobian matrix's maximum and minimum singular values. Admittedly, this calculation is not straightforward, and using the Frobenius norm formulation is recommended, as shown in (3).

$$\kappa = \frac{1}{n}\sqrt{tr(JNJ^T)tr(J^{-1}NJ^{-T})} \in [1, \infty) \tag{3}$$

where n is the matrix dimension of the Jacobian J, tr is the trace, and N is given by (4), where I is the identity matrix.

$$N = \frac{1}{n} I_{n\times n} \tag{4}$$

The condition number can be interpreted in different ways depending on the author. The distance to a kinematic singularity or the relationship between output force and workspace velocity from input torques and joint velocities is among the most common [39]. An index value close to unity is the best possible value for the above-mentioned conditions. It shows that the singular values of the Jacobian matrix are equal, and the manipulator is said to be in an isotropic configuration. The condition number is not fully bounded (3); to prevent the index from taking large values, its reciprocal is used, known as the Local Condition Index (LCI) defined in (5).

$$LCI = \frac{1}{\kappa} \in [0, 1] \tag{5}$$

The LCI is a local performance index; it depends on the current configuration of the robot at the time of calculation. The extension of LCI to the entire workspace is detailed in [37], a simpler computational calculation is proposed in [40] and is given as:

$$GCI = \frac{1}{n_{WS}} \sum_{j=1}^{n_{WS}} \frac{1}{\kappa} \in (0, 1). \tag{6}$$

where n_{WS} is the number of workspace nodes. The GCI (6) is completely bounded and estimates the distribution of κ in the entire workspace.

D. GENETIC ALGORITHMS

Genetic algorithms are the most popular technique in evolutionary computing and are used to solve optimization problems in different fields of science. They are based on genetics and Darwin's theory of evolution, whereby, for a specific task, the fittest individual survives through a process of natural selection [41].

The main stages of GA can be described in a simplified form:

- **Start**: a population of n individuals is generated.
- **Selection**: the best individuals (parents) from the population are selected based on their problem-solving ability (fitness). It is common for the best individuals to be chosen.
- **Reproduction**: based on a crossover probability, offspring (children) are obtained from the parents. With a probability of mutation, children can suffer mutations in their genes.
- **Evaluation**: the fitness of the new children are evaluated.
- **Replacement**: the past population is replaced with new individuals, obtaining a new generation of candidates to solve the problem.
- **Stop**: if the stopping condition is satisfied or the number of generations is reached; otherwise, return to the selection step.

GA have different methods to carry out each of the stages mentioned above; because of their extension, it is recommended to look at [41] for a detailed review. For this work, a basic GA is used.

E. PARTICLE SWARM OPTIMIZER

The Particle Swarm Optimizer algorithm (PSO) presented in [42] is a technique based on the social behavior presented by groups of animals (flock of birds or a school of fish) or insects (colony or swarm of ants, bees, or termites) [41]. Each individual is defined as a particle within the swarm and behaves influenced by their intelligence and the swarm (collective) intelligence. The behavior of the swarm to achieve a specific task is based on the communication between particles. When a particle finds a favorable condition, the swarm redirects its behavior to follow it. The movement of the swarm is based on the position and velocity attributes of each particle.

The algorithm that describes the behavior of the standard PSO can be summarized as [43]:

1) Suppose a swarm of N particles looking for the minimum of the function $f : \mathbb{R}^n \to \mathbb{R}$ in k iterations.
2) The position and velocity of the particle are defined as $x_i \in \mathbb{R}^n$ and $v_i \in \mathbb{R}^n$, with $i = 1, \ldots, N$.
3) Denote p_{bi} as the best solution for the particle i and g_b the global best solution.
4) In the first iteration $k = 0$. For $i = 1, \ldots, N$, generate random positions $x_i^{(0)}$ and velocities $v_i^{(0)}$.
5) Set $p_{bi}^{(0)} = x_i^{(0)}$.
6) Set $g_b^{(0)} = \arg\min_{x \in \{x_i^{(0)}, \ldots, x_N^{(0)}\}} f(x)$.
7) Then generates random n-vectors $r_i^{(k)}$ and $s_i^{(k)}$ with random numbers evenly distributed over a range of (0, 1).
8) Calculate the new velocity for the particles from the past values as:

$$v_i^{(k+1)} = w v_i^{(k)} + c_1 r_i^{(k)} \circ (p_{bi}^{(k)} - x_i^{(k)}) + c_2 s_i^{(k)} \circ (g_b - x_i^{(k)}). \quad (7)$$

9) Calculate the position of the new particles using (7):

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}. \quad (8)$$

10) Using (8), if $f(x_i^{(k+1)}) < f(p_{bi}^{(k)})$, calculate the new personal best $p_{bi}^{(k+1)} = x_i^{(k+1)}$, else $p_{bi}^{(k+1)} = p_{bi}^{(k)}$.
11) Using (8), if $f(x_i^{(k+1)}) < f(g_b^{(k)})$, calculate the new global best $g_b^{(k+1)} = x_i^{(k+1)}$, else $g_b^{(k+1)} = g_b^{(k)}$.
12) Stop if the value of the particles converges or the stop condition is reached. Otherwise $k = k + 1$, then return to step 7.

From (7) the operator $\circ$ stand for Hadamard product, w is defined as an inertial constant, c_1 and c_2 represent cognitive (personal) and social (group) learning rates. These affect how much the p_{bi} and g_i parameters influence the displacement of the particle. The recommended values for these parameters are $w \in [0.4, 0.9]$, and $c_1, c_2 \approx 2$.

For this work, a standard PSO algorithm is used.

F. GREY WOLF OPTIMIZER

The Grey Wolf Optimizer algorithm (GWO) presented in [23] is based on the behavior of gray wolves when hunting in packs, their hierarchical organization, and social interaction. In a pack of wolves, these can be classified depending on the level of the hierarchy; consider the three most essential wolves: the leader of the pack is the alpha wolf (α), and the second and third wolves are denoted as beta (β) and delta (δ) respectively. The remaining wolves are referred to as omega (ω). To replicate the hunting process of the wolf pack, we make the following assumptions:

- Suppose a population (pack) of size N solution candidates (wolfs) for an optimization problem.
- Like wolves, the population of solutions can be divided into the categories α,β,δ, and ω.
- Wolves stalk their prey until they encircle it (Fig. 2). To do this, they move randomly around the prey, helped by the positions of the wolves with the highest hierarchy (α,β,δ).
- When the prey stops, the wolves attack.

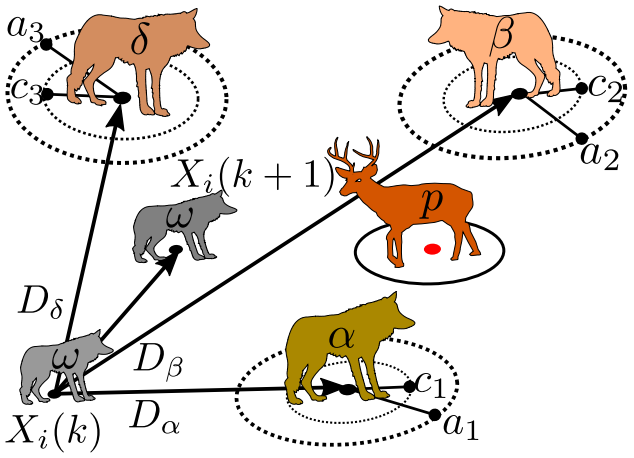

FIGURE 2. Positioning of the gray wolves around its prey and its possible next locations in a two-dimensional space.

The mathematical method that mimics the behavior of wolves to encircle and hunt their prey is given by the following equations:

$$D = |CX_p(k) - X_i(k)|, \tag{9}$$
$$X_i(k+1) = X_p(k) - AD, \tag{10}$$
$$A = 2ar_1 - a, \tag{11}$$
$$C = 2r_2. \tag{12}$$

where k is the number of iterations, in (9) D represent the distance between the prey $X_p(k)$ and the i wolf $X_i(k)$, with $i = 1, \ldots, N$. For the wolf next position, (10), A and C are coefficient vectors. The vector A conditions the hunting of the prey ($A > 1$) and its search ($A < 1$). The vector C mimics obstacles to approaching prey in nature. Finally (11) and (12), the terms r_1 and r_2 are random values in [0, 1], the term a is linearly decreased from 2 to 0 as iterations run.

To model the wolf hunting method, we do not know the position of the prey (solution of the problem); we assume that the α, β, and δ wolves have better knowledge about it and are taken as a reference for the ω wolves.

$$D_\alpha = |C_1X_\alpha(k) - X_i(k)| \tag{13}$$
$$D_\beta = |C_2X_\beta(k) - X_i(k)| \tag{14}$$
$$D_\delta = |C_3X_\alpha(k) - X_i(k)| \tag{15}$$

With the distances calculated regarding the wolves α (13), β (14), and δ (15), new reference positions are calculated.

$$X_1(k) = X_\alpha(k) - A_1D_\alpha \tag{16}$$
$$X_2(k) = X_\beta(k) - A_2D_\beta \tag{17}$$
$$X_3(k) = X_\delta(k) - A_3D_\delta \tag{18}$$

Using (16)-(18) the next values for the ω wolfs positions are calculated.

$$X_i(k+1) = \frac{X_1 + X_2 + X_3}{3} \tag{19}$$

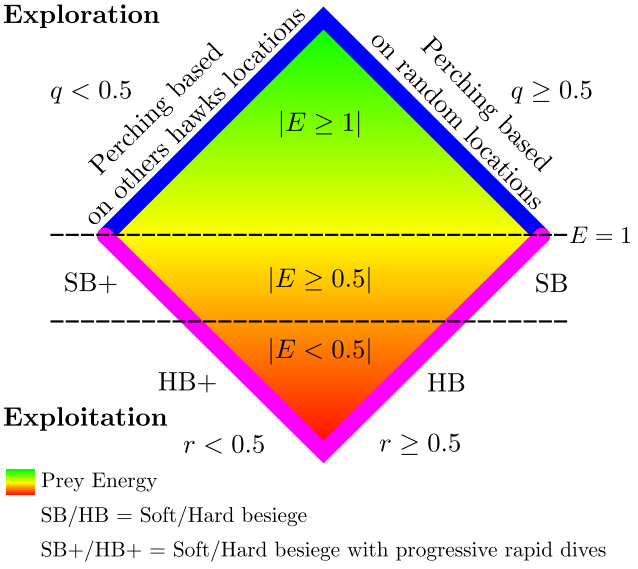

FIGURE 3. Exploration and exploitation phases of HHO algorithm.

With the equations described previously, the GWO algorithm can be summarized as follows:

1) Initialize a size N gray wolf population and the A, C parameters.
2) Evaluate the fitness of all the wolves. Then select the best three results for α, β, and δ wolves.
3) Loop until the maximum number of iterations is reached.
   - Loop for updating the position of each wolf in the population using (19).
   - Then updates A and C.
   - Calculate the new fitness for all the wolves. Then select the best three results and update α, β, and δ wolves.
   - Increase iteration.
4) Return α wolf as the best solution.

For this work, the basic GWO algorithm introduced above is used.

G. HARRIS HAWK OPTIMIZATION

The optimization algorithm presented in [30] is inspired by the cooperative behavior of Harris's Hawks when hunting prey in an escape situation. The hunting tactic comprises a series of surprise pounces made by several hawks from different directions. As in nature, this algorithm also considers the possibility that the prey escapes the attack and is chased until it exhausts.

According to Fig. 3, the HHO algorithm can be summarized as:

For the exploration stage:

- Consider the energy the prey has to escape as E.
- When the prey has high energy levels $E \geq 1$, the exploration phase depends on the factor q.

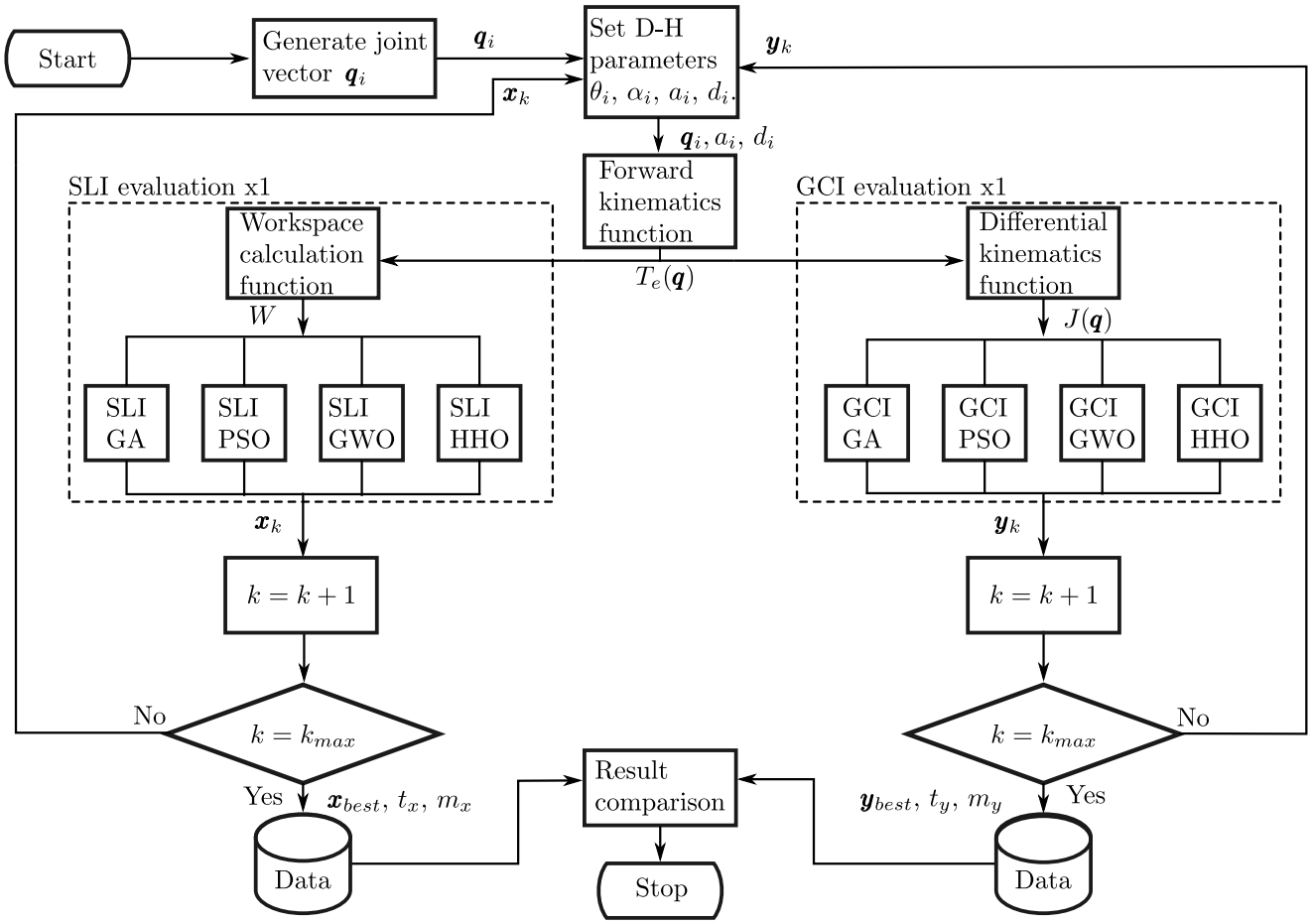

FIGURE 4. Methodology block diagram for robot arm optimization.

- If $q \geq 0.5$, the hawks are perched in random positions. Otherwise, they are located based on the position of other hawks ($q < 0.5$).

Hawks move from the exploration to the hunting phase as the energy level of the prey drops. For the exploitation stage:

- Assume the term r as the escape probability of the prey. Consider $r \geq 0.5$ as a failed escape. Otherwise, $r < 0.5$ for a successful escape.
- The energy level and the probability of failed escape are used to determine if the hawks besiege the prey in a soft ($r \geq 0.5$ and $E \geq 0.5$) or the hard way ($r \geq 0.5$ and $E < 0.5$).
- If the probability of successful escape is high ($E \geq 0.5$ and $r < 0.5$), the hawks perform soft besieges with progressive rapid dives to correct their position and direction to the deceptive movements made by the prey when escaping.
- Finally, when $E < 0.5$, the prey does not have enough energy to escape, the falcons prepare a series of hard besieges with progressive rapid dives before finally catching the prey (find the best solution).

The mathematical models of the above concepts can be reviewed in-depth in [30].

For this work, a standard HHO algorithm is used.

III. METHODOLOGY

The diagram of the methodology proposed for this work is presented in Fig. 4, and a description of the processes is detailed below.

From the manipulator presented in Fig. 1, it is proposed to optimize the dimensions of the main links a_2, d_4, and d_6 using SLI and GCI as objective functions.

Firstly, the vector of joint variables $\boldsymbol{q_i}$ is generated for each joint of the robot with $i = 1, \ldots, 6$. The size of the vector depends on the desired resolution of the indices to be evaluated (SLI and GCI). The vector $\boldsymbol{q_i}$ sets the position and orientation of the robot, and it is generated using the beta distribution method proposed in [44]. The objective of this method is to obtain a better distribution of joint values and improve the calculation of the workspace of the robot arm. The vector $\boldsymbol{q_i}$ is calculated only once at the beginning of each optimization problem so that the four algorithms have the same joint values for their respective k iterations.

Second, the DH parameters (Table 1) are completed with the vector $\boldsymbol{q_i}$ containing the values for θ_i and with the values of the vector $\boldsymbol{x}_k$ for SLI or $\boldsymbol{y}_k$ for GCI. It is assumed that the vector $\boldsymbol{x}_k$ contains the solution values generated by the

optimization technique (GA, GWO, PSO, or HHO). In this way, the vector $\boldsymbol{x}_k$ contains the values of the lengths a_2, d_4 and d_6, which finished defining the parameters of the robot in all the positions generated previously with the vector $\boldsymbol{q}_i$.

Third, the forward kinematics function is calculated using the homogeneous transformation matrices method [36] for all the configurations generated in the previous step. Forward kinematics function generate a $T_e(\boldsymbol{q})$ matrix (20) composed of a rotation matrix R_e (robot orientation) and a robot position vector $\boldsymbol{P}_e$.

$$T_e(\boldsymbol{q}) = \begin{bmatrix} R_e & \boldsymbol{P}_e \\ 0^T & 1 \end{bmatrix} \tag{20}$$

For the left side of the diagram, with the generated $T_e(\boldsymbol{q})$ matrices, the function that calculates the workspace of the robot is generated. The method proposed in [44] is used for this. It is based on a cloud of points and its subsequent division into horizontal and vertical sections from which the area is calculated and later joined to obtain an approximation of the volume of the point cloud. For this function W, the value of the vector $\boldsymbol{x}_k$, which contains the lengths of the robot, and the values of the $T_e(\boldsymbol{q})$ matrices are necessary. With the Workspace function ready, it is used by each optimization technique to calculate the SLI defined in (1). This index is used as an objective function to optimize the proposed algorithms GA, PSO, GWO, and HHO. These generate a vector of solutions $\boldsymbol{x}_k$ with which the operations of the previous blocks are repeated for each iteration within the optimization algorithm until the total number of generations k.

Consider $\boldsymbol{x} = [x_1, x_2, x_3]$ as the solution vector for the optimization problem (21).

$$\min_{\boldsymbol{x}\in[0.1,1.0]} \frac{L(\boldsymbol{x})}{\sqrt[3]{W(\boldsymbol{x})}} \tag{21}$$

Then the minimization problem is subject to the following nonlinear constraints (22) - (25):

$$g_1 = \frac{x_1}{x_2} \geq 1.1, \tag{22}$$

$$g_2 = \frac{x_1}{x_2} \leq 2.0, \tag{23}$$

$$g_3 = \frac{x_2}{x_3} \geq 1.1, \tag{24}$$

$$g_4 = \frac{x_2}{x_3} \leq 2.0. \tag{25}$$

Finally, when the number of generations is reached ($k = k_{max}$), the values of the vector $\boldsymbol{x}_k$ of the last iteration of the algorithm are saved together with the values of the time and memory occupied during the execution of the algorithm, this is repeated with each of the four optimization techniques.

For the right side of the diagram, the process continues from the Forward kinematics function and the transformation matrices $T_e(\boldsymbol{q})$, proceeding to perform a function that calculates the differential kinematics of the robot and obtains the Jacobian matrix $J(\boldsymbol{q})$ for each of the configurations generated in the previous processes. Subsequently, the Jacobian matrix is used to operate the GCI index defined in (6) and is used as an objective function for each optimization algorithm. For the right side, it is assumed that each optimization technique generates a vector of solutions $\boldsymbol{y}_k$, which contains the values of the robot links with which the DH parameters are completed, and the processes mentioned above are repeated. This sequence continues until the maximum number of generations is reached.

Define $\boldsymbol{y} = [y_1, y_2, y_3]$ as the solution vector for the following optimization problem (26):

$$\max_{\boldsymbol{y}\in[0.1,1.0]} \frac{1}{n_{WS}} \sum_{j=1}^{n_{WS}} \frac{1}{\kappa(\boldsymbol{y})}. \tag{26}$$

Like the SLI minimization problem (21), the GCI maximization problem (26) is subject to equivalent nonlinear constraints (27)-(30).

$$g_5 = \frac{y_1}{y_2} \geq 1.1, \tag{27}$$

$$g_6 = \frac{y_1}{y_2} \leq 2.0, \tag{28}$$

$$g_7 = \frac{y_2}{y_3} \geq 1.1, \tag{29}$$

$$g_8 = \frac{y_2}{y_3} \leq 2.0. \tag{30}$$

The inequality constraints for $\boldsymbol{x}_k$ and $\boldsymbol{y}_k$ are represented by g_i with $i = 1, \ldots, 8$, the values for these constrains are proposed in [45]. They are based on the link length ratio of commercial manipulator robot arms.

The methodological diagram process can be summarized as follows, for each optimization problem (21) and (26) in each k-th iteration within the GA, PSO, GWO and HHO algorithms, a solution vector ($\boldsymbol{x}_k$ or $\boldsymbol{y}_k$) is generated with the optimized link lengths of the robot arm a_2, d_4, and d_6. When the proposed number of generations is reached, each algorithm delivers the best convergence value for the corresponding optimization problem ($\boldsymbol{x}_{best}$ and $\boldsymbol{y}_{best}$), besides from the computation time (t_x and t_y) and the memory (m_x and m_y) used in the execution of the algorithm.

Finally, the results obtained from each optimization algorithm are compared to determine their performance in the problem of optimal design of serial robotic arms. The aspects to be evaluated in the comparison are the computation time, the number of iterations to obtain a convergent solution and the optimization results for the proposed robot.

For implementing this methodology, the following considerations were taken:

- All four algorithms are implemented with their standard values.
- Then the value of d_1 is set to 30cm.
- The range of joint variables in radians is set: q_1 and $q_5 = [-\pi/2, \pi/2]$, q_2 and $q_3 = [-\pi/3, 2\pi/3]$, finally for q_4 and $q_6 = [-\pi, \pi]$.
- A million points are used to calculate the workspace volume of the robot for the SLI problem.
- 5000 points are used to calculate the GCI problem.

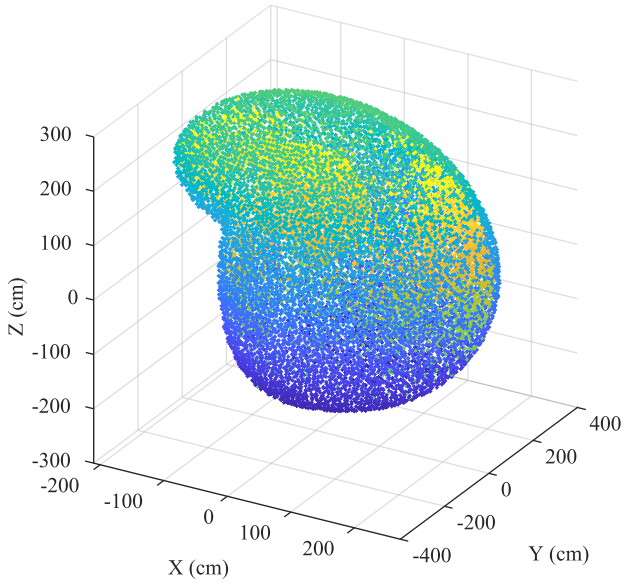

FIGURE 5. Workspace volume of the robot arm optimized by HHO and SLI.

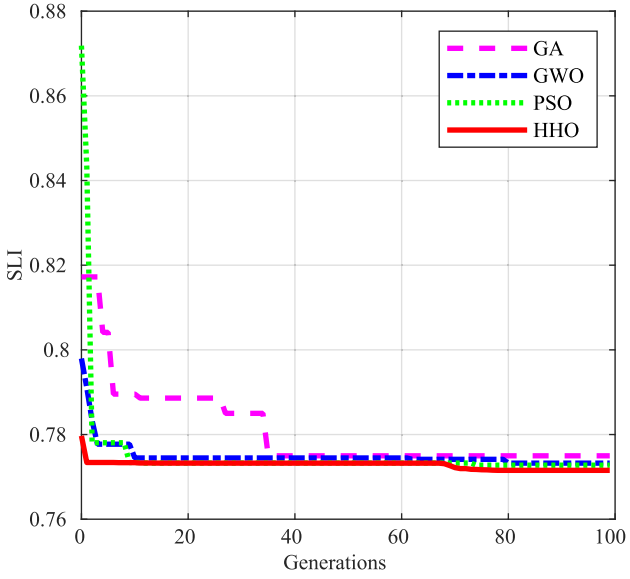

FIGURE 6. Convergence curves of the optimization algorithms for the SLI index.

Subsequently, the algorithms used are initialized with the following values:

- GA: population size: 50, generations: 100, tournament size: 5, mutation and crossover rates : 0.25, tournament selection, uniform crossover, uniform mutation.
- PSO: population size: 50, iterations: 100, c_1 and c_2: 2, and w: 0.7.
- GWO: population size: 50, and iterations: 100.
- HHO: population size: 50, iterations: 100, and levy factor: 0.01.

Implementing the performance indexes is done in the Python programming language version 3.9, and the optimization algorithms are implemented from the Python micro-framework NiaPy for bio-inspired algorithms [46].

The function (26) is based on the equation (3), and it is evaluated several times for each individual in the population. Due to the high number of evaluations of the same function, it is executed in parallel, dividing the process between the cores of the computer. Then, each optimization algorithm evaluates the GCI function individually.

The characteristics of the computer equipment used in this implementation are presented below.

- CPU: Intel Xeon, 8 × 2.8GHz.
- RAM: 8 GB.
- Storage: 10 GB.
- OS: Ubuntu 18.04.
- Provider: Google Cloud.

The results obtained through the different optimization techniques are reviewed and discussed below.

IV. RESULTS AND DISCUSSION

The objective of SLI is to find the best relationship between the length of the links and a larger workspace of the robot through a low index value. In this sense, a robot with the largest workspace volume is desirable.

TABLE 2. Results of the optimization for the SLI.

	a_2 (cm)	d_4 (cm)	d_6 (cm)	W (cm^3)	SLI
GA	98.73	88.57	78.51	55.61×10^6	0.7749
PSO	100	90.85	81.08	59.64×10^6	0.7722
GWO	100	90.88	80.21	59.03×10^6	0.7732
HHO	100	90.90	82.64	60.89×10^6	**0.7715**

TABLE 3. Computational resources and performance of optimization algorithms for SLI.

	SLI_σ	Time (s)	Memory (MB)
GA	0.010100	18024	304
PSO	0.012077	17984	296
GWO	0.003169	18016	296
HHO	**0.001030**	25190	297

For the SLI minimization problem, the results obtained for the lengths of links, workspace, and index value are shown in Table 2. Table 3 shows the standard deviation of convergence of the algorithms, the computation time, and the memory occupied in the implementation problem. Furthermore, Fig. 6 shows the convergence curves of each optimization algorithm.

According to Table 2, we observe that the proposed algorithms yield solutions that tend to the upper limit of the range of the solution vector. Also, we observe that all the algorithms converge to similar values. However, the best result for the index is given by the HHO algorithm. The HHO algorithm presents an SLI advantage over GA, PSO, and GWO over 0.44%, 0.09%, and 0.22%, respectively.

In this context, although the difference between the algorithms is small, the workspace values have significant differences.

The value of the largest workspace corresponds to the robot generated with HHO (Fig. 5); the advantage between the volumes generated with the other algorithms is 3% for GWO, 2% for PSO, and 8.6% for GA, this being the volume with the lowest value. The fastest converging algorithm is HHO, followed by GWO, PSO, and finally GA.

The values of the standard deviation in Table 3 help review the behavior of each algorithm regarding their convergence values. A low value shows that the search region of the algorithm is better because the value of the first approximation is close to the final optimal value with fewer fluctuations. Based on Table 3, we observe that the search region of the HHO is the best due to it having the lowest standard deviation. HHO reaches a stable value in the second approximation, which is close to the best solution obtained in iteration 98. For PSO, the standard deviation is higher because of the high value in its first approximation. This algorithm achieves a stable value at iteration 10 and its best approximation at iteration 74. The GA is the algorithm that converges first, but its staggering behavior and the value of its first approximation mean that it needs 35 iterations to obtain a stable value. The convergence value of the other algorithms is better for the same iteration. The GWO algorithm reaches a stable value in iteration 11 and converges in iteration 80. In addition, the value of convergence through time is between PSO and GA. Finally, from the first iteration, we observe that HHO presents an SLI value equal to or better than that provided by the other algorithms.

Regarding the computation time, PSO presented the best value, but there are no significant differences between it and GA or GWO. However, between PSO and HHO, we observed a 40% increase in time. Respecting the memory used in each implementation, we found no significant differences.

The GCI optimization results obtained for the lengths of the links, workspace, and index value are shown in Table 4. In Fig. 7, the convergence curves for each algorithm are shown. Table 5 shows the standard deviation of convergence of the algorithms, computation time, and memory occupied in the maximization problem.

The optimization of the GCI comprises finding the lengths for the links of the robot that deliver the index with a value closer to the unit, representing that the robot is better conditioned. Taking the Yaskawa robot arm shown in Fig. 1(a) as reference, and replacing the DH values from Table 1: $d_1 = 33$ cm, $d_4 = 40.5$ cm, $d_6 = 8$ cm, $a_1 = 8.8$ cm, $a_2 = 40$ cm, $a_3 = 4$ cm, and bounding the vector $\boldsymbol{q}_i$ in the range of $[-\pi, \pi]$ to use the beta distribution method with $q_1 = [-17\pi/18, 17\pi/18]$, $q_2 = [-13\pi/36, 5\pi/6]$, $q_3 = [-32\pi/45, 5\pi/4]$, $q_5 = [-3\pi/4, 3\pi/4]$, and q_4, $q_6 = [-\pi, \pi]$.

When implementing the same methodology for calculating GCI, we obtain a value of 0.18258, with a maximum value for LCI of 0.5826. According to the values shown in Table 5, we observe that the robot link lengths tend toward the lower limit of the range of the solution vector. The optimization of the index entails the reduction of the length links, and consequently, the robot workspace volume is reduced. Additionally, the average value for GCI is 0.21359 and 0.9285 for LCI. Therefore, we observe an improvement of 14% for GCI and 37% for LCI regarding the robot arm mentioned above.

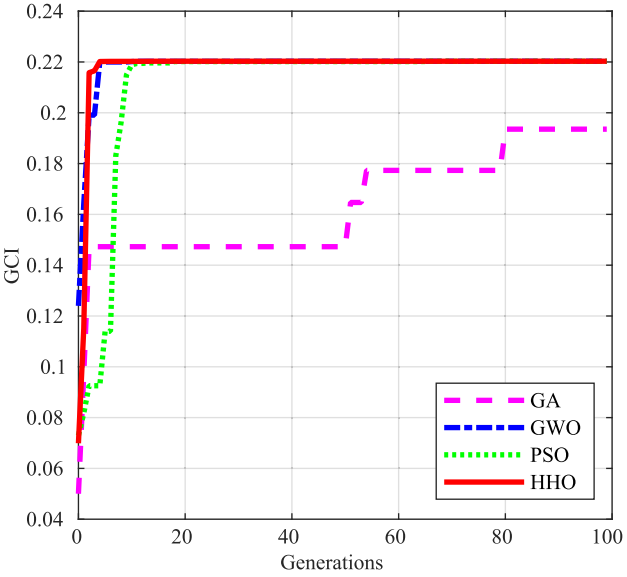

FIGURE 7. Convergence curves of the optimization algorithms for the GCI index.

TABLE 4. Results of the optimization for the GCI.

	a_2 (cm)	d_4 (cm)	d_6 (cm)	W (cm^3)	GCI	LCI
GA	17.56	15.41	11.36	257.16×10^3	0.19354	0.8111
PSO	16.68	13.53	10	190.39×10^3	0.22026	0.9658
GWO	16.51	13.23	10	184.06×10^3	0.22028	**0.9689**
HHO	15.57	13.32	10	186.02×10^3	**0.22029**	0.9682

TABLE 5. Computational resources and performance of optimization algorithms for GCI.

	GCI_σ	Time (s)	Memory (MB)
GA	0.0231	32576	95
PSO	0.0325	33374	95
GWO	**0.0114**	32022	94
HHO	0.0182	57025	95

Focusing on the performance of optimization algorithms, GA obtains the convergence value farthest from the desired value. Whereas the PSO, GWO, and HHO algorithms converge to practically the same value, the best result for the GCI index corresponds to the HHO.

The robot resulting from optimization with HHO shows an improvement in GCI of 0.012% compared to PSO, 0.002% for GWO, and 12% compared to GA. Even though that HHO has the best value for GCI, the best value of LCI is obtained with GWO. The reason is that while for a joint configuration, the lengths of the robots can improve the value of the index, for other configurations, the index can be affected.

Considering the volume of the workspace obtained with HHO as a reference, it shows a 2.3% decrease for PSO and

38.2% for GA. However, it presents an improvement in the volume of 1% concerning GWO. In this regard, we note that the difference in volume between GA and HHO is due to the optimized link size delivered by each algorithm.

Regarding convergence speed, when reviewing Fig. 7, we observe that the slowest algorithm is GA, followed by PSO and GWO, while HHO is the fastest converged algorithm. When reviewing the standard deviation of Table 5, we observe that the minimum value corresponds to the GWO, so it has the best search region. Since it reaches a stable value in iteration 5, its best value is obtained in iteration 36 and remains constant until the end of the run. Although we observe in the graph a fast convergence of HHO, its standard deviation is not the best because of the low value of its first approximation and the fact that it makes subtle adjustments from iteration 5, where it reaches a convergent value until it reaches its best result in iteration 94. The standard deviation of PSO is the highest because it takes ten iterations to reach the reference value, and it makes adjustments over time until it reaches its final value at iteration 97. Finally, the GA presents a standard deviation value between PSO and GWO because of a slow and stepped convergence with its best value in iteration 80.

Regarding computing time, the fastest implementation corresponds to GWO, with an advantage of 1.7% over GA, 4.2% for PSO, and 78% over HHO. As in the previous SLI problem, there are no significant differences in memory in implementing GCI techniques.

An additional test is carried out to study the robustness of the optimization algorithms. This test consists of doing a multi-run evaluation for GCI and SLI functions. For the tests, the following configurations of the algorithms were made.

- In order to reduce the time of the simulations and to test the robustness of the algorithms, a reduction in the population and iterations of the optimization algorithms are carried out.
- The population size was reduced from 50 to 20 individuals (40%) for all algorithms for both SLI and GCI.
- The number of generations was reduced from 100 to 20 (20%) for SLI and GCI.
- Workspace reduced from 1 million to 300,000 positions (30%) for SLI and from 5,000 to 1,500 positions (30%) for GCI.
- 25 tests were performed for each algorithm.

The test results for SLI are shown in Fig 8. We observe in the boxplot the robustness of the algorithms for 25 runs. The box plot displays the most relevant information about a data set and provides an overview of the data distribution and its symmetry. Based on the above, it can be seen that the PSO algorithm shows the best behavior. However, the best value for the index is obtained with HHO even though this technique has a less consistent behavior than PSO. The second-best performance is GWO, which has less data variability than HHO and GA. Regarding GA, a more dispersed behavior of the results is shown, and its minimum value is far from those obtained by GWO, PSO, and HHO.

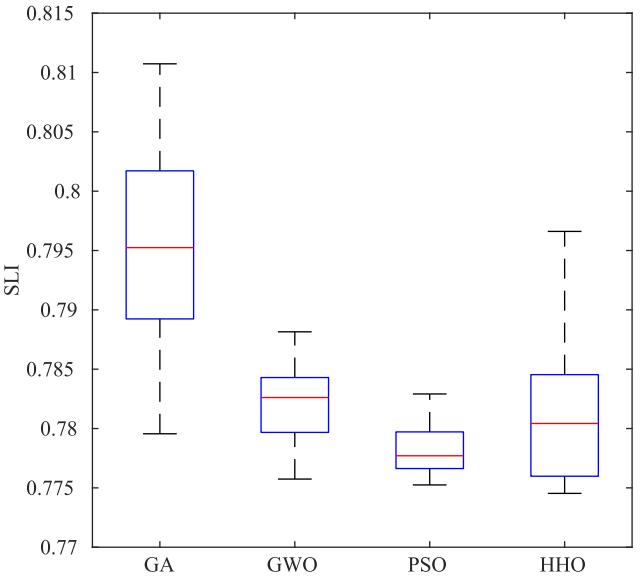

FIGURE 8. Result of the 25-run test of the optimization algorithms for the SLI index where PSO is the most robust technique.

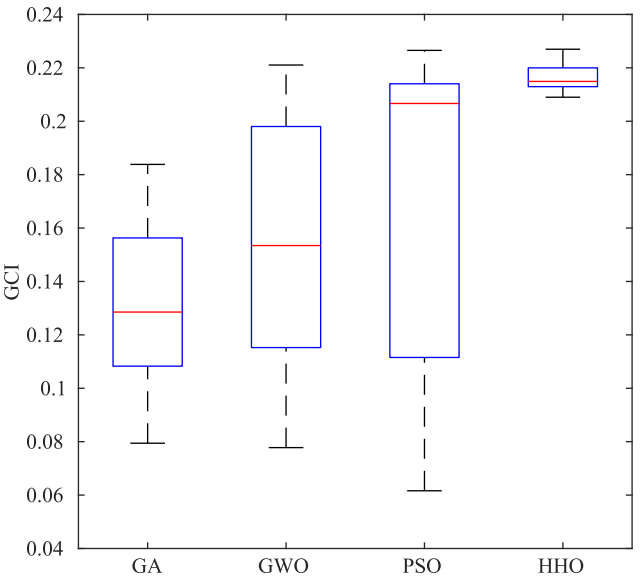

FIGURE 9. Result of the 25-run test of the optimization algorithms for the GCI index where HHO is the most robust technique.

The results obtained for the 25-run test for GCI are shown in Fig 9. In the boxplot, we observe that the best behavior of the optimization algorithms is obtained with HHO, obtaining the best value for the GCI index among all the tests followed by the maximum value of PSO. The second best behavior is obtained by GA, while PSO and GWO have dispersed behaviors since their range of values is wider than HHO. In the case of GA and GWO, an almost symmetrical distribution of their values is observed, while for PSO, a distribution loaded towards values far from the maximum value is found.

Finally, it is observed that the distribution of HHO tends towards the maximum value.

V. CONCLUSION

This work proposes bio-inspired algorithms HHO and GWO to address the optimal design problem of a 6DOF serial robot arm using SLI and GCI indices as objective functions. Then a performance comparison against GA and PSO is realized.

For the SLI problem, the robot obtained with HHO has a large workspace volume because of the large link lengths. On the other hand, for the GCI problem, the resulting robot improves the index value of the Yaskawa robot and also achieves a position close to an isotropic configuration. However, the workspace volume obtained is smaller than the volume of the Yaskawa Motoman-MH5LS II robot.

The comparison shows that HHO, PSO, and GWO obtain similar results in GCI and SLI optimization problems. Nevertheless, HHO obtains the best results with a high convergence speed, despite having the longest computational time. GWO has a performance close to PSO and a high convergence speed, but GWO has the best search region. GA has the lowest convergence speed and obtains a result similar to the other techniques for the SLI problem, whereas the result is the furthest from the convergence value for GCI.

Regarding the 25-run test for the minimization of SLI, an improvable behavior of the HHO algorithm is shown, with PSO being the algorithm with minor variability in the results. However, the behavior of the other proposed algorithm GWO is acceptable compared to HHO and GA. In the case of the maximization of GCI, it is found that HHO has the best behavior, followed by GA, GWO, and PSO, but their range of values is considerably more extensive, so they have a more dispersed behavior.

The behavior of the algorithms for these optimization problems can be influenced by the sensitivity to the population size of each technique, in addition to a reduced number of generations. Another factor to consider is the resolution of the functions to be evaluated; since these have only 30% of the original size of the evaluated data, this can be influenced by the joint values of the robot that are randomly generated using beta distribution.

Considering the low number of iterations needed to get a value close to the final optimal result and that it presents a better convergence value through each iteration respecting the other algorithms, HHO is a recommended optimization technique for the optimal design problem of robotic arms.

The comparison presented in this work can help designers to reduce the time needed to obtain an optimal design for different manufacturing problems.

In future works, implementing the proposed algorithms in multi-objective optimization problems for different types of robots is considered. Besides, we propose exploring other bio-inspired algorithms, their modifications, and their application in the optimal design of robots.

REFERENCES

[1] A. J. Kurdila and P. Ben-Tzvi, *Dynamics and Control of Robotic Systems*. Hoboken, NJ, USA: Wiley, 2019.

[2] A. Filippeschi, P. Griffa, and C. A. Avizzano, "Kinematic optimization for the design of a collaborative robot end-effector for tele-echography," *Robotics*, vol. 10, no. 1, p. 8, Jan. 2021.

[3] S. Zarkandi, "Kinematic analysis and workspace optimization of a novel 4RPSP+PS parallel manipulator," *Mech. Based Des. Struct. Mach.*, vol. 49, no. 1, pp. 131–153, 2021.

[4] M. Russo, S. Herrero, O. Altuzarra, and M. Ceccarelli, "Kinematic analysis and multi-objective optimization of a 3-UPR parallel mechanism for a robotic leg," *Mechanism Mach. Theory*, vol. 120, pp. 192–202, Feb. 2018.

[5] T. Sun and B. Lian, "Stiffness and mass optimization of parallel kinematic machine," *Mechanism Mach. Theory*, vol. 120, pp. 73–88, Feb. 2018.

[6] S. Kucuk, "Dexterous workspace optimization for a new hybrid parallel robot manipulator," *J. Mech. Robot.*, vol. 10, no. 6, Dec. 2018, doi: 10.1115/1.4041334.

[7] Y. Zheng, Y. Wang, and J. Liu, "Research on structure optimization and motion characteristics of wearable medical robotics based on improved particle swarm optimization algorithm," *Future Gener. Comput. Syst.*, vol. 129, pp. 187–198, Apr. 2022.

[8] M. Nabipour and S. A. A. Moosavian, "Human model in the loop design optimization for RoboWalk wearable device," *J. Mech. Sci. Technol.*, vol. 35, no. 10, pp. 4685–4693, Oct. 2021.

[9] J. Huo, B. Yang, H. Ru, and J. Huang, "Parametric design optimization of a universal supernumerary robotic limb," in *Proc. Int. Symp. Micro-NanoMehatronics Hum. Sci. (MHS)*, Nagoya, Japan, Dec. 2021, pp. 1–6.

[10] Z. Qin, Z. Liu, Y. Liu, H. Gao, C. Sun, and G. Sun, "Workspace analysis and optimal design of dual cable-suspended robots for construction," *Mechanism Mach. Theory*, vol. 171, May 2022, Art. no. 104763.

[11] S. Hwang, H. Kim, Y. Choi, K. Shin, and C. Han, "Design optimization method for 7 DOF robot manipulator using performance indices," *Int. J. Precis. Eng. Manuf.*, vol. 18, no. 3, pp. 293–299, Mar. 2017.

[12] Q. Meng, J. Li, H. Shen, J. Deng, and G. Wu, "Kinetostatic design and development of a non-fully symmetric parallel delta robot with one structural simplified kinematic linkage," *Mech. Based Des. Struct. Mach.*, pp. 1–21, Jun. 2021.

[13] J. Amar and K. Nagase, "Genetic-algorithm-based global design optimization of tree-type robotic systems involving exponential coordinates," *Mech. Syst. Signal Process.*, vol. 156, Jul. 2021, Art. no. 107461.

[14] D. M. Bodily, T. F. Allen, and M. D. Killpack, "Multi-objective design optimization of a soft, pneumatic robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 1864–1871.

[15] A. Zeiaee, R. Soltani-Zarrin, R. Langari, and R. Tafreshi, "Kinematic design optimization of an eight degree-of-freedom upper-limb exoskeleton," *Robotica*, vol. 37, no. 12, pp. 2073–2086, Dec. 2019.

[16] H. Dong, E. Asadi, C. Qiu, J. Dai, and I.-M. Chen, "Geometric design optimization of an under-actuated tendon-driven robotic gripper," *Robot. Comput.-Integr. Manuf.*, vol. 50, pp. 80–89, Apr. 2018.

[17] E. G. López, W. Yu, and X. Li, "Optimum design of a parallel robot using neuro-genetic algorithm," *J. Mech. Sci. Technol.*, vol. 35, no. 1, pp. 293–305, Jan. 2021.

[18] X. Li, K. Sun, C. Guo, T. Liu, and H. Liu, "Design, modeling and characterization of a joint for inflatable robotic arms," *Mechatronics*, vol. 65, Feb. 2020, Art. no. 102311.

[19] A. Schmitz, P. Berthet-Rayne, and G.-Z. Yang, "Endoscopic bi-manual robotic instrument design using a genetic algorithm," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Macau, China, Nov. 2019, pp. 2975–2982.

[20] M.-F. Leung, C. A. C. Coello, C.-C. Cheung, S.-C. Ng, and A. K.-F. Lui, "A hybrid leader selection strategy for many-objective particle swarm optimization," *IEEE Access*, vol. 8, pp. 189527–189545, 2020.

[21] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, Oct. 2021.

[22] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, vol. 10, pp. 10031–10061, 2022.

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

[24] L. Chen, Y. Zhang, Y. Xue, and Y. Chen, "Robot path planning based on improved particle swarm optimization," in *Proc. Power Syst. Green Energy Conf. (PSGEC)*, Aug. 2022, pp. 887–891.

[25] C. Choubey and J. Ohri, "Optimal trajectory generation for a 6-DOF parallel manipulator using grey wolf optimization algorithm," *Robotica*, vol. 39, no. 3, pp. 411–427, Mar. 2021.

[26] R.-E. Precup, E.-I. Voisan, R.-C. David, E.-L. Hedrea, E. M. Petriu, R.-C. Roman, and A.-I. Szedlak-Stinean, *Nature-Inspired Optimization Algorithms for Path Planning and Fuzzy Tracking Control of Mobile Robots*. Singapore: Springer, 2021, pp. 129–148.
[27] Y. Wu, B. Ren, and H. Chen, "Application of grey wolf optimization in anti-collision vehicle," in *Proc. 5th Int. Conf. Mach. Learn. Soft Comput.*, New York, NY, USA, Jan. 2021, pp. 135–140, doi: 10.1145/3453800.3453825.
[28] R. G. Roy and D. Ghoshal, "Grey wolf optimization-based second order sliding mode control for inchworm robot," *Robotica*, vol. 38, no. 9, pp. 1539–1557, Sep. 2020.
[29] S. Dereli, "A new modified grey wolf optimization algorithm proposal for a fundamental engineering problem in robotics," *Neural Comput. Appl.*, vol. 33, no. 21, pp. 14119–14131, Nov. 2021.
[30] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future Gener. Comput. Syst.*, vol. 97, pp. 849–872, Aug. 2019.
[31] M. Issa and A. Samn, "Passive vehicle suspension system optimization using Harris hawk optimization algorithm," *Math. Comput. Simul.*, vol. 191, pp. 328–345, Jan. 2022.
[32] E. H. Houssein, M. Ahmad, M. E. Hosney, and M. Mazzara, *Classification Approach for COVID-19 Gene Based on Harris Hawks Optimization*. Cham, Switzerland: Springer, 2021, pp. 575–594.
[33] B. Li, Z. Bian, and Z. Wang, "Research on the electric life prediction of relay contact based on Harris hawk optimized," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.
[34] M. Wu, D. Yang, and T. Liu, "Harris hawks optimization for solving flexible jobshop scheduling problem considering energy consumption," in *Proc. 6th Int. Conf. Robot. Autom. Eng. (ICRAE)*, Nov. 2021, pp. 388–392.
[35] D. Izci, S. Ekinci, A. Demirören, and J. Hedley, "HHO algorithm based PID controller design for aircraft pitch angle control system," in *Proc. Int. Congr. Hum.-Comput. Interact., Optim. Robotic Appl. (HORA)*, Jun. 2020, pp. 1–6.
[36] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control* (Advanced Textbooks in Control and Signal Processing). London, U.K.: Springer 2010.
[37] C. Gosselin and J. Angeles, "A global performance index for the kinematic optimization of robotic manipulators," *J. Mech. Des.*, vol. 113, no. 3, pp. 220–226, 1991.
[38] J. K. Salisbury and J. J. Craig, "Articulated hands: Force control and kinematic issues," *Int. J. Robot. Res.*, vol. 1, no. 1, pp. 4–17, Mar. 1982.
[39] S. Patel and T. Sobh, "Manipulator performance measures-a comprehensive literature survey," *J. Intell. Robotic Syst.*, vol. 77, no. 3, pp. 547–570, 2015.
[40] L. J. Puglisi, R. J. Saltaren, H. A. Moreno, P. F. Cárdenas, C. Garcia, and R. Aracil, "Dimensional synthesis of a spherical parallel manipulator based on the evaluation of global performance indexes," *Robot. Auto. Syst.*, vol. 60, no. 8, pp. 1037–1045, Aug. 2012.
[41] S. S. Rao, *Modern Methods of Optimization*. Hoboken, NJ, USA: Wiley, 2009, ch. 13, pp. 693–736.
[42] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.
[43] E. K. Chong and S. H. Zak, *Global Search Algorithms*. Hoboken, NJ, USA: Wiley, 2013, ch. 14, pp. 273–301.
[44] Y. Cao, K. Lu, X. Li, and Y. Zang, "Accurate numerical methods for computing 2D and 3D robot workspace," *Int. J. Adv. Robotic Syst.*, vol. 8, no. 6, p. 76, Dec. 2011.
[45] S. Kucuk and Z. Bingul, "Comparative study of performance indices for fundamental robot manipulators," *Robot. Auton. Syst.*, vol. 54, no. 7, pp. 567–573, Jul. 2006.
[46] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., "NiaPy: Python microframework for building nature-inspired algorithms," *J. Open Source Softw.*, vol. 3, no. 23, p. 613, Mar. 2018, doi: 10.21105/joss.00613.

ERVIN GALAN-URIBE received the B.S. degree in robotics engineering from the Polytechnic University of Guanajuato, Guanajuato, Mexico, in 2016, and the M.S. degree in mechatronics from the Autonomous University of Queretaro, Santiago de Querétaro, Mexico, in 2018, where he is currently pursuing the Ph.D. degree in mechatronics. His research interests include robotics, signal processing, and optimization algorithms.

LUIS MORALES-VELAZQUEZ (Member, IEEE) received the Ph.D. degree in mechatronics from the Autonomous University of Queretaro, Mexico, in 2010. He is currently a National Researcher Level 1 with the Mexican Council of Science and Technology, CONACYT. He is also a Professor with the Autonomous University of Queretaro. He is the author of more than 30 technical papers published in international journals and conferences. His research interests include hardware signal processing with FPGA and monitoring and diagnosis on dynamic systems.