Review Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
September-October 2018: 1–17

DOI: 10.1177/1729881418801132
journals.sagepub.com/home/arx
SAGE

Metaheuristic techniques comparison to optimize robotic end-effector behavior and its workspace

Miguel Angel Funes Lora[1], Edgar Alfredo Portilla-Flores[2], Raul Rivera Blas[1], Emmanuel Alejandro Merchán Cruz[1] and Manuel Faraón Carbajal Romero[1]

Abstract

Many robots are dedicated to replicate trajectories recorded manually; the recorded trajectories may contain singularities, which occur when positions and/or orientations are not achievable by the robot. The optimization of those trajectories is a complex issue and classical optimization methods present a deficient performance when solving them. Metaheuristics are well-known methodologies for solving hard engineering problems. In this case, they are applied to obtain alternative trajectories that pass as closely as possible to the original one, reorienting the end-effector and displacing its position to avoid the singularities caused by limitations of inverse kinematics equations, the task, and the workspace. In this article, alternative solutions for an ill-posed problem concerning the behavior of the robotic end-effector position and orientation are proposed using metaheuristic algorithms such as cuckoo search, differential evolution, and modified artificial bee colony. The case study for this work considers a three-revolute robot (3R), whose trajectories can contain or not singularities, and an optimization problem is defined to minimize the objective function that represents the error of the alternative trajectories. A tournament in combination with a constant of proportionality allows the metaheuristics to modify the gradual orientation and position of the robot when a singularity is present. Consequently, the procedure selects from all the possible elbow-configurations the best that fits the trajectory. A classical numerical technique, Newton's method, is used to compare the results of the applied metaheuristics, to evaluate their quality. The results of this implementation indicate that metaheuristic algorithms are an efficient tool to solve the problem of optimizing the end-effector behavior, because of the quality of the alternative trajectory produced.

Keywords

Robotic manipulator, metaheuristics, singularity, path, inverse kinematics, cuckoo search, differential evolution, artificial bee colony, Newton's method

Date received: 10 January 2017; accepted: 23 August 2018

Topic: AI in Robotics; Human Robot/Machine Interaction
Topic Editor: Andrey V Savkin
Associate Editor: M Reza Emami

Introduction

Singularities during robotic motion are an important problem when using robots, and its solution is an open issue due to its degree of complexity and the extended use of such devices. Those singularities can be classified as internal or boundary. Boundary singularities occur when the manipulator is stretched or retracted and are easy to avoid if the robot is not taken to its workspace limits. Internal

[1] Instituto Politécnico Nacional-SEPI-ESIME UA, Av, Mexico, MX
[2] Instituto Politécnico Nacional-CIDETEC, Av. "Juan de Dios Bátiz" s/n, Mexico, MX

Corresponding author:
Miguel Angel Funes Lora, Instituto Politécnico Nacional SEPI-ESIME UA, Av. de las Granjas No 682, Santo Tomas, 02020, Mexico 02250, MX.
Emails: miguelaf@umich.edu; apertura_dr@hotmail.com

singularities occur within the available workspace, and they are caused by the alignment of two or more axes, or by the end-effector configuration. In these singularities, the end-effector position and/or orientation produces undesired movements in one or more specific points during a trajectory. Unlike the first type of singularities, these constitute a serious problem since they can appear in any part of the robot envelope when a trajectory in the operational space has been generated.[1]

A way to solve the problem is by detecting in advance the singularities in a specific trajectory, and the subsequent calculation of a new optimized path from the set of points corresponding to the original trajectory. Nevertheless, the problem has not been addressed from this avoidance perspective. There are several techniques for solving manipulator kinematics as an optimization problem such as neural networks, hybrid algorithms, Jacobian approximation, and dual quaternions, among others.[2–8] In the study by Srisuk et al.,[9] a custom neural network implementation is used to solve the inverse kinematics of a 4 DOF robot with the ability to reach objects in 3-D avoiding obstacles within its envelope. In the study by Horváth et al.,[10] an interesting solution is presented using the Jacobian pseudo-inverse and the best mobility direction. Another recent robotic-algorithm implementation is the length relaxation method.[11] This methodology consists on determining an approximation kinematic chain with randomly located points, considering that the links are not identical, and the extreme positions are known (base and end-effector). As a difference to other approaches, it focuses its attention on obtaining the Cartesian coordinates located in the joint intersection and consequently, the angles of every joint.

In addition to the aforementioned methodologies, alternative approaches such as metaheuristics have been applied. They are approximated techniques for solving hard problems established as optimization cases. These algorithms are used in different engineering areas such as mechanism synthesis,[12] the design of electromagnetic devices,[13] delivery planning,[14] network routing,[15] among others. In this work, a novel methodology for avoiding robotic singularities in a trajectory is proposed, by establishing an optimization problem with a weighted objective function that modifies the position and/or orientation in the Euclidean space, enabling the manipulator to maintain its reference direction concerning to the original target. Cuckoo search (CS), differential evolution (DE), and modified artificial bee colony (MABC) were implemented for this development, using two specific functions as test cases with and without singularities.

The content of this article is as follows: In the second section, the end-effector kinematics, manipulator elbow-configurations, and the equations that define the paths are presented. In the third section, the objective function and the selection criteria for elbow-configuration are described, while in the fourth section the procedure for optimizing trajectories and the implemented metaheuristic algorithms are explained in detail. The fifth section includes the experimental results and their analysis. Finally, in the sixth section, the conclusions are discussed.

Forward kinematics and manipulator elbow-configuration

In order to accomplish the optimization proposed in this article, it is required to establish the parameters that conform the kinematic chain of the manipulator. The Denavit–Hartenberg parameters are used to obtain a 4×4 matrix as a representation of the position and orientation of a rigid body.[16] Two parameters describe the dimension of every link (d, a) and two more (α, θ) correspond to the angles generated in every joint.[17] For the implemented experimentation in a 3R robot (see Figure 1), the links have the following parameters: $a_1 = 1$, $a_2 = 1$, $a_3 = 0.6$, θ_1, θ_2, and θ_3; additionally $d_i = 0$ and $\alpha_i = 0$, for $i = 1, 2$, and 3. Substituting these parameters, it is possible to obtain the homogeneous transformation matrix of each of the corresponding links, A_2^1 and A_3^1, in equations (1) and (2), respectively:

$$A_2^1 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & a_1\cos(\theta_1)+a_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & a_1\sin(\theta_1)+a_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$A_3^1 = \begin{bmatrix} xu\vartheta & yu\vartheta & zu\vartheta & Px\vartheta \\ xv\vartheta & yv\vartheta & zv\vartheta & Py\vartheta \\ xw\vartheta & yw\vartheta & zw\vartheta & Pz\vartheta \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

A_3^1 represents the forward kinematics representation of the robot, where:

$$\begin{aligned} xu\vartheta &= \cos(\theta_1+\theta_2+\theta_3) \\ xv\vartheta &= \sin(\theta_1+\theta_2+\theta_3) \\ xw\vartheta &= 0 \\ yu\vartheta &= -\sin(\theta_1+\theta_2+\theta_3) \\ yv\vartheta &= \cos(\theta_1+\theta_2+\theta_3) \\ yw\vartheta &= 0 \\ zu\vartheta &= 0 \\ zv\vartheta &= 0 \\ zw\vartheta &= 1 \end{aligned}$$

$$Px\vartheta = a_1\cos(\theta_1) + a_2\cos(\theta_1+\theta_2) + a_3\cos(\theta_1+\theta_2+\theta_3)$$

$$Py\vartheta = a_1\sin(\theta_1) + a_2\sin(\theta_1+\theta_2) + a_3\sin(\theta_1+\theta_2+\theta_3)$$

$$Pz\vartheta = 0$$

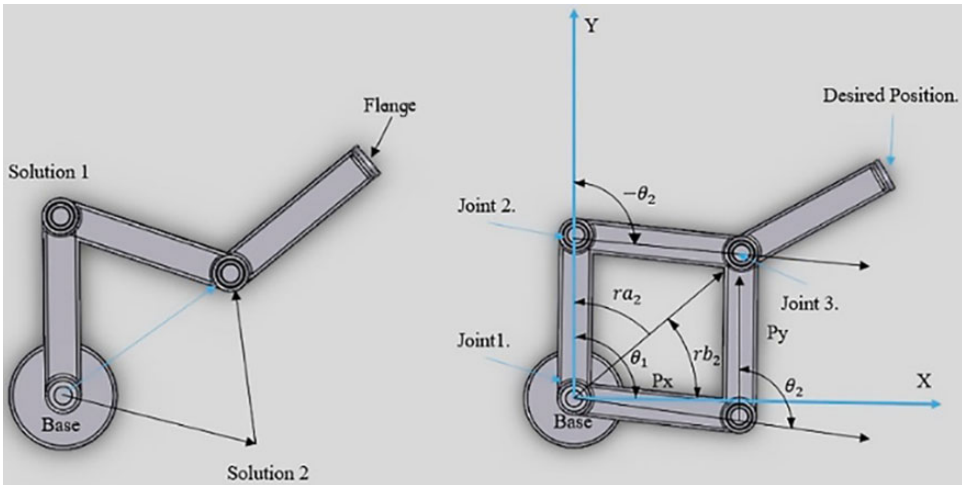

Figure 1. Multiple solutions to the 3R robot configuration and their corresponding parameters.

A planar 3R robot arm contemplates a large workspace because any position within the interior of its envelope can be reached.[18] However, there is a singularity problem that arises depending on the numerical methodology and the number of elbow-configurations in which a robot can reach the target with a desired position and orientation. As can be seen in Figure 1, the manipulator has multiple elbow-configurations, which involves sequencing the system to select the optimal for every run, based on the best solution recorded from the last iteration.

The equations corresponding to the mathematical development of the elbow-configuration from Figure 1 are as follow: assuming that $\theta_2 = -\theta_2^*$,[17] the images of each of the angles can be determined by means of the homogeneous transformation matrix from A_2^1; thus, the elements (1,4) and (2,4) of A_2^1 are represented in equations 3 and 4, and the sum of its squares is as in equation (5).

$$Px = a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2) \tag{3}$$

$$Py = a_1 \sin(\theta_1) + a_2 \sin(\theta_1 + \theta_2) \tag{4}$$

$$\kappa = a_1^2\Big(\sin^2(\theta_1) + \cos^2(\theta_1)\Big)$$

$$\xi = a_2^2\Big(\sin^2(\theta_{12}) + \cos^2(\theta_{12})\Big)$$

$$\tau = \Big(2a_1a_2\Big(\cos(\theta_1)\cos(\theta_1+\theta_2) + \sin(\theta_1)\sin(\theta_1+\theta_2)\Big)\Big)$$

$$Px^2 + Py^2 = \kappa + \xi + \tau \tag{5}$$

Considering equations from (6) to (11), it is possible to obtain equation (12)

$$Px^2 + Py^2 = a_1^2 + a_2^2 + 2a_1a_2\cos(\theta_2) \tag{6}$$

$$\theta_2 = atan2\left(\frac{+/-\sqrt{1-\delta^2}}{\delta}\right) \tag{7}$$

$$\delta = \frac{P_x^2 + P_y^2 - a_1^2 - a_2^2}{2a_1a_2} \tag{8}$$

$$\nu = atan2\left(\frac{Py}{Px}\right) \tag{9}$$

$$\psi = atan2\left(\frac{a_2\ sin(\theta_2)}{a_1 + a_2\ cos(\theta_2)}\right) \tag{10}$$

$$\theta_1 = \nu - \psi \tag{11}$$

$$\theta_3 = \theta_{123} - \theta_1 - \theta_2 \tag{12}$$

The inverse kinematics can be obtained using closed-form solutions or numerical methodologies. The former allows a quick and efficient calculation of the elbow-configuration once the desired end-effector position and orientation is determined. From a numerical perspective, the inverse kinematics can be formulated to determine the articulatory coordinates $X = [\theta_1, \theta_2, \theta_3]^{\mathrm{T}}$ as configurations of a given manipulator, depending on the formulation and the existence of multiple solutions.[19] Recalling that the implementation is based on the orientation discrimination it is necessary to know the Cartesian points and the orientation elements by means of a homogeneous transformation matrix that defines the trajectory sequence. As shown in equations (13) and (15), these functions represent the path that the manipulator follows in x and y coordinates of a sinusoidal and a spiral function, where equations (14) and (16) represent their final position and orientation of the end-effector.

$$f_{1\sin}(\chi) = \frac{1}{100}\Big(10\ \sin(5\chi)\Big) + 1\ \ and\ f_{2\sin}(\chi) = \frac{1}{100}\chi \tag{13}$$

The orientation and translation for the sinusoidal function are defined by the range [−160, 160] with increments of

$\Delta = 1$, between its limits, obtaining the normal orientation of the plane η, where $\Upsilon_1 = [\chi \in \mathbb{R} : -160 \leq \chi \leq 160]$.

$$M_{\sin} = \begin{bmatrix} \cos(\eta) & -\sin(\eta) & 0 & f_{1\sin}(\chi) \\ \sin(\eta) & \cos(\eta) & 0 & f_{2\sin}(\chi) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

$$\phi = \left(-0.1e^{-10\chi} \sin(\sqrt{9900}\chi + 1.5) \right)$$

$$\varpi = \left(0.9950\, e^{-10\chi} \cos(\sqrt{9900}\chi + 1.5) \right)$$

$$f_{1\text{sp}}(\chi) = 0.8(\phi + \varpi) \text{ and } f_{2\text{sp}}(\chi) = 1.5 + 0.8\left(e^{-10\chi} \sin(\sqrt{9900}\chi + 1.5) \right) \quad (15)$$

The orientation and translation for the spiral function are defined by the range [0, 1] with increments $\Delta = 2.7778\, e^{-3}$ between its limits, where $\Upsilon_2 = [\chi \in \mathbb{R} : 0 \leq \chi \leq 1]$.

$$M_{\text{spiral}} = \begin{bmatrix} 1 & 0 & 0 & f_{1\text{sp}}(\chi) \\ 0 & 1 & 0 & f_{2\text{sp}}(\chi) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (16)$$

It is important to mention that M_{spiral} has 360 poses recorded in an interval [0,1], and $M_{\sin}$ has 320 poses in the interval [−160,160], and every recorded pose corresponds to one optimization problem.

The objective function

The primary purpose of a mono-objective optimization problem is to find the minimum or maximum of a function (optimal). Since an optimum is a solution, it can be local, global, or both.[20,21] In this work, the objective function focuses on reducing the error between the desired position and orientation, and the best obtained with a metaheuristic, using the elements of equation (2) that constrain the kinematic chain. For this case, $\theta_1, \theta_2,$ and θ_3 are the design variables, establishing that

$$X = [x_1, x_2, x_3]^{\mathrm{T}} = [\theta_1, \theta_2, \theta_3]^{\mathrm{T}}$$

Must minimize the objective function, $f(X)$. This function is conformed as in equation (17)

$$\min f(X) = \text{function1} + \text{function2} + \text{function3} + \text{function4} + \text{function5} + \text{function6} \quad (17)$$

$$X \in R^3$$

where

$$\text{function1} = \left(\left(a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2) + a_3 \cos(\theta_1 + \theta_2 + \theta_3) \right) - Px_3^1 \right)^2$$

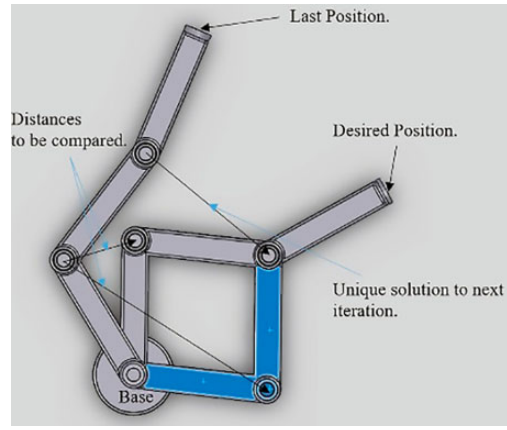

Figure 2. Difference of distances between the previous position and the possible current elbow-configuration.

$$\text{function2} = \left(\left(a_1 \sin(\theta_1) + a_2 \sin(\theta_1 + \theta_2) + a_3 \sin(\theta_1 + \theta_2 + \theta_3) \right) - Py_3^1 \right)^2$$

$$\text{function3} = \left(\cos(\theta_1 + \theta_2 + \theta_3) - xu_3^1 \right)^2 k$$

$$\text{function4} = \left(\sin(\theta_1 + \theta_2 + \theta_3) - xv_3^1 \right)^2 k$$

$$\text{function5} = \left(-\sin(\theta_1 + \theta_2 + \theta_3) - yu_3^1 \right)^2 k$$

$$\text{function6} = \left(\cos(\theta_1 + \theta_2 + \theta_3) - yv_3^1 \right)^2 k$$

The design variables are subject to expression (18)

$$-\text{Lower Limit} \leq \theta_i \leq \text{Upper Limit } \forall\ i = 1, 2, 3 \quad (18)$$

where k defines a relative weight between position and orientation as well as a direct relationship with the maximum angle of end-effector inclination. The lesser the k value, the greater the penalization rank for $[0 \leq k \leq 1]$. Due to the global search properties of the metaheuristics, it is necessary to determine if there are multiple configurations exist (see Figure 2). Otherwise, elbow-configuration solutions can vary from one iteration to another, making it necessary to determine which of them is the optimal, not only for its aptitude but also for its configurability.

Figure 3 shows how the global search finds solutions during the $M_{\sin}$ trajectory. In general, metaheuristics do not exceed the limits of the design variables, unlike Local Search Methods (LSM) such as NM, which do not need to be bounded. After finding all possible elbow-configurations with their corresponding values for,θ_1, θ_2, and θ_3 it is essential to consider the Euclidean norm between the last solution executed by the robot and the new one for each possible configuration. For this

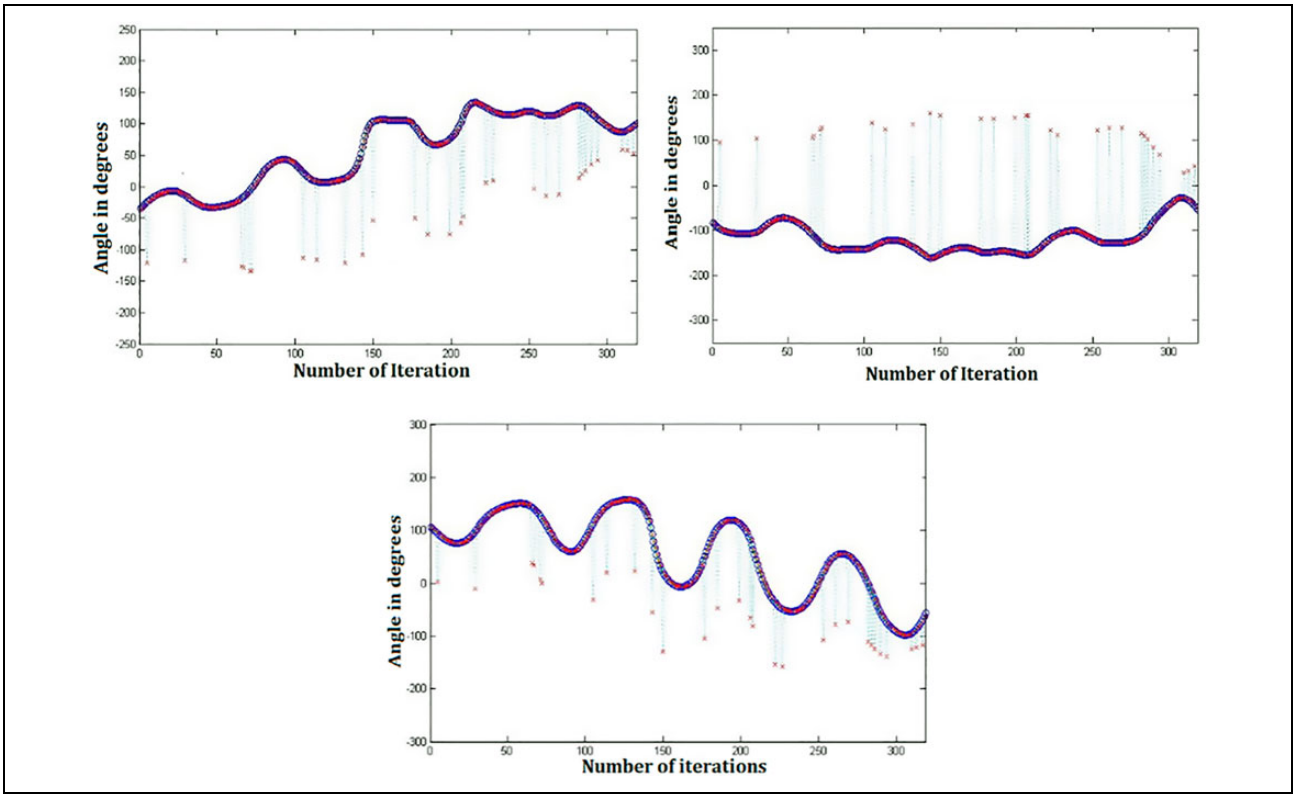

Figure 3. Example of the robot angles $j_1, j_2,$ and j_3 for $k = 0.2$ in the DE algorithm. DE: differential evolution.

specific case only two norms are analyzed, (see equations (19) and (20)):

$$fd_1 = \sqrt{\left(Att_0(1,4) - Att_{0P}(1,4)\right)^2 + \left(Att_0(2,4) - Att_{0P}(2,4)\right)^2} \tag{19}$$

$$fd_2 = \sqrt{\left(Att_{0r2}(1,4) - Att_{0P}(1,4)\right)^2 + \left(Att_{0r2}(2,4) - Att_{0P}(2,4)\right)^2} \tag{20}$$

where Att_0 and Att_{0P} represent the actual and previous Cartesian positions related to joint 1 j_1, and Att_{0r2} is the second elbow-configuration coordinate of the same joint. The optimal solution or the best solution found is determined comparing the closest distance between Att_0 and Att_{0r2} with respect to Att_{0P}. As can be seen in Figure 3, the crosses represent the original trajectory obtained by the metaheuristic, and the circles indicate the final path. Once k approaches to one, the angular tolerance is reduced producing disruptions in the elbow-configuration. This inconvenience also can occur in conventional numerical methods due to their high susceptibility to the starting point and local search characteristics.

Procedure for optimizing the test trajectories

In general, robot arms cannot take decisions independently, this means that generally there is an operator in charge of the movements produced by the manipulator. However, it is possible for some robots to suggest alternative solutions with the aim to optimize a process or a specific task, in which the user has the chance to decide the application of such optimization. One of the advantages that offer the implemented metaheuristics is the capability to produce solutions based on decision-making by means of a greedy selection. Imagine that a person wants to catch an object with a desired orientation and position. If at least one of the two parameters cannot be reached, the person tries to stretch out his arm to intercept the target, discriminating one of the two parameters. Under this criterion, three cases should be considered in the manipulator decision-making: existence, uniqueness, and stability.[22] In Figure 4, an overview of how the robot operates to solve the ill-posed problem is shown. The first block (I) represents the human–machine interaction (HMI) and the variable declaration. In the second block (II), the path generation box determines the trajectory depending on the operator preference. Once the path is declared, the optimization stage (in orange) will optimize the robot inverse kinematics for every point traced by considering the proposed objective function. This stage corresponds to each of the algorithms implemented in this article (DE, CS, MABC, and NM). Finally, the simulation is ready to solve the dynamics and control in the same manner as the HMI feedback. However, this work is limited to the singularity avoiding approach.

The classical optimization methods are analytical, and their objective is to find the optimal solution of continuous

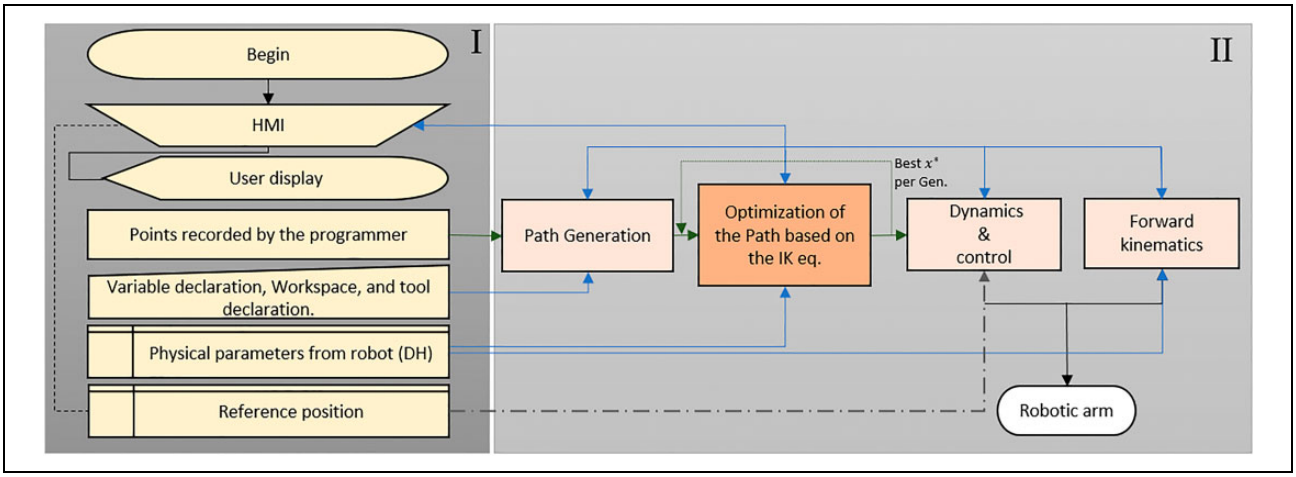

Figure 4. Flowchart diagram of the robot data management scheme.

Algorithm 1. Differential evolution/rand/1/bin

1. Initialize population randomly, where: $x := (x_1, \dots, x_d)^T$.
2. Add the best solution stored by the last robot movement.
3. Evaluate the objective function $f(x)$
4. for m=1 to GenMax
5. Select three individuals randomly $x_{r0,g}, x_{r1,g}, x_{r2,g}$ by $v_{i,g} = x_{r0,g} + F(x_{r1,g} - x_{r2,g})$
6. Analyze the bounds for $v_{i,g}$
7. for k=1 to var, (Crossover)
8. if $rand(0,1) \leq CR$ or $j = j_{rand}$
9. $u_{i,g} = v_{i,g}$
10. else
11. $u_{i,g} = x_{i,g}$
12. end if
13. Evaluate the objective function $f(x)$
14. end for
15. if $u_{i,g} \leq x_{i,g}$ (Selection of trial or target vector)
16. $x_{i,g+1} = u_{i,g}$
17. else
18. $x_{i,g+1} = x_{i,g}$
19. end if
20. end for
21. Add the best solution found in the next population of step 1 for the following pose.

and differentiable functions.[23] However, these methodologies have a limited approach because some practical problems involve objective functions that do not fulfill such characteristics. Then, it is necessary to use alternative methods such as metaheuristics, which are consistent algorithms for optimization problems, based on some explicit search strategy in the set of all feasible solutions .[24] The metaheuristics used in this article are nature-inspired algorithms (NIA) that can be classified by tournament type[25,26] and their interaction with LSM.[27–31] In general, the optimization problems are classified as monobjective or multiobjective,[32–35] also taking into account the type of constraints,[36] and if they are combinatory or numerical.

Evolutionary algorithms are highly effective metaheuristics that have been applied in diverse types of engineering problems.[25,37,38] In the case of DE, the algorithm begins with the creation of an initial population in which a target vector and a base vector are selected. Then, some individuals are chosen randomly and extracted from the population to create an entirely new individual; the next step is crossing the result with a target solution, and this creates a new trial vector. The result is compared with the target, in a tournament by greedy selection. This procedure is repeated for a specific number of generations declared as GenMax until convergence to an optimal or suboptimal is obtained.[25,32,38] The pseudo-code of the implemented DE version is in Algorithm 1.

Algorithm 2. Cuckoo search.

```
1.  Initialize randomly a population of n host nests, x_n := (x_1, ..., x_d)^T
2.  Evaluate the objective function f(x_n)
3.  Keep the best solution
4.  while i < GenMax
5.      Select one nest randomly and the best of every iteration
6.      Generate a new solution based on the two nests by random walk, checking the design
        variable limits
7.      Evaluate the objective function from the new solution
8.      if  F_new ≤ F_rand
9.          Replace the random nest with the new solution
10.     end if
11.     if  rand(0,1) ≤ p_a
12.         Find the worse nest and replace this with one using as a reference the same nest
             verifying limits
13.         Evaluate the objective function from the new solution
14.     end if
15.     Keep the best solution from the generation
16. end while
17. Add the best solution found in the population of step 1 or the best solution stored by the last
    robot movement
```

The second implemented metaheuristic is the CS algorithm that operates simulating the cuckoo breeding parasitism (see Algorithm 2). Cuckoos lay eggs in different nests randomly (one egg per nest). The nests with better eggs (optimal solutions) pass to the next generation. The host also has the chance to discover a foreign egg, and it can either discard the egg or abandon the nest and build a new one somewhere else.[39–41] CS runs up to a maximum number of generations GenMax, until one solution is reached. An important thing about CS is that this algorithm can work with the help of an additional local search method such as the random walk algorithm.

The MABC is a NIA, based on the foraging behavior of honey bees.[36] MABC uses three types of bees for the implementation: employee, onlooker, and scout, to find new food sources (solutions). Once a bee reaches a source, it flies to another nearby, and both food sources are compared in a greedy selection. Later, onlooker bees seek for new solutions from the ones with the best fitness. When a food source has not been improved after a specific number of iterations, then it is replaced by one scout bee, generating a new random solution.[42,43] Again, the maximum number of generations GenMax terminates the algorithm. The MABC pseudo-code can be seen in Algorithm 3.

The last algorithm implemented is a numerical methodology called Newton's method (NM), which uses the pseudo-inverse Jacobian to get faster convergence in comparison with metaheuristics (assuming that a solution exists.) From NM approach, it is possible to solve nonlinear simultaneous equations starting from the fact that they can be linearized using Taylor expansion.[44] Two stop criteria were implemented for NM: (1) Max$N$ is a constant for finishing the algorithm once a specific number of iterations have been reached, and (2) an additional constant called ε is in charge of terminating the algorithm if the absolute solution of every subfunction in OF is less than ε itself, (see Algorithm 4).

It is also important to mention that OF is the difference between the actual and the desired value, considering both position and orientation during a specific pose in the trajectory. In other words, OF is the minimization of the difference between the desired and the best solution obtained. Consequently, the second stop criterion is as follows

$$\begin{aligned}\text{if } &(|f_1| < \varepsilon \,\&\, |f_2| < \varepsilon \,\&\, |f_3| < \varepsilon \,\&\, |f_4| < \varepsilon \\ &\&\, |f_5| < \varepsilon \,\&\, |f_6| < \varepsilon \,\&\, |f_7| < \varepsilon \,\&\, |f_8| < \varepsilon \\ &\&\, |f_9| < \varepsilon \,\&\, |f_{10}| < \varepsilon \,\&\, |f_{11}| < \varepsilon \,\&\, |f_{12}| < \varepsilon)\end{aligned}$$

Experimental results and discussions

Three metaheuristic algorithms (DE, CS, and MABC) were implemented to evaluate the proposed objective function performance in two different trajectories, during 60 runs. The first trajectory is generated by a spiral function, in which 360 points are recorded. Every recorded point corresponds to one different optimization problem. The second trajectory is derived from a sinusoidal function, in which 320 points are recorded.

The first algorithm studied, DE, was tuned as recommended in previous works.[25] In this context, it is possible to get an initial estimation for the DE parameters by considering the expressions (21) to (23), where the range of the mutation scale factor F is $[0.5,\ 1.0]$, with a crossover rate interval of $[0.8,\ 1.0]$

Algorithm 3. Modified artificial bee colony.

1. Initialize the set of food sources randomly, where: $x_i^0, i = 1, \ldots, SN$.
2. Evaluate the objective function $f(x)$ of each $x_i^0, i = 1, \ldots, SN$.
3. Add the best solution stored by the last robot pose.
4. while g < GenMax
5. for $i = 1$ to SN
6. Generate sources v_i^g using x_i^{g-1} and x_{Best}^{g-1} $such\ as$: $v_i^g = x_{Best}^{g-1} + \phi_j\left(x_i^{g-1} - x_k^{g-1}\right)$
7. Evaluate v_i^g and verify limits.
8. if v_i^g is better than x_i^{g-1}, then:
9. $x_i^g = v_i^g$
10. $limit_i = 0$
11. else
12. $limit_i = limit_i + 1$
13. end if
14. end for
15. for $i = 1$ to SN
16. Select, based on fitness proportional selection food source and the best individual of the population x_l^g.
17. Generate v_l^g with x_l^g and x_{Best}^{g-1}.
18. Evaluate v_l^g and verify limits.
19. if v_l^g is better than x_l^g
20. $x_l^g = v_l^g$
21. $limit_i = 0$
22. else
23. $limit_i = limit_i + 1$
24. end if
25. end for
26. Generate new food sources randomly for those whose limit to improve has been reached and keep the best solution.
27. end while
28. Add the best solution found in the following population of step 1.

Algorithm 4. Newton's method.

1. Set the last position recorded by the robot as a reference
2. for $n = 1$ to $MaxN$
3. Consider the initial estimate of the solution as: $x_j = \hat{x}_j + \Delta x_j$ for $j = (1, \ldots, 3)$
4. Evaluate the Jacobian and the original function (Equation 17)
5. Evaluate the value of $ds = J \backslash f^T$ and $x^* = \hat{x}_j + ds_i$
6. if the absolute solution of every function is less than the tolerance called ε, then stop the NM algorithm
7. end if
8. end for

$$F_{\text{dither}} = F_l + \text{rand}_g\ (0,1)\ (F_h - F_l) \tag{21}$$

$$\text{Cr}_{\text{dyn}} = \text{Cr}_l + \text{rand}_g\ (0,1)\ (\text{Cr}_h - \text{Cr}_l) \tag{22}$$

$$N_p = 10 \cdot D \tag{23}$$

For the proposed trajectories, the tuning of DE was as follows: $D = 3$ (design variables), $N_p = 30$ (population size), and GenMax = 500 (maximum number of generations). Figures 5(c) and 6(c) show the convergence of the objective function, while Figures 5(d) and 6(d) present a zoomed view of such convergence after approximately 200 generations during the entire spiral and sinusoidal trajectories.

For the Cuckoo Search implementation, the tuning criterion is according to Xin-She Yang and Suasg Deb,[41] the number of nest should be in between $n = 14$ and 40, and the

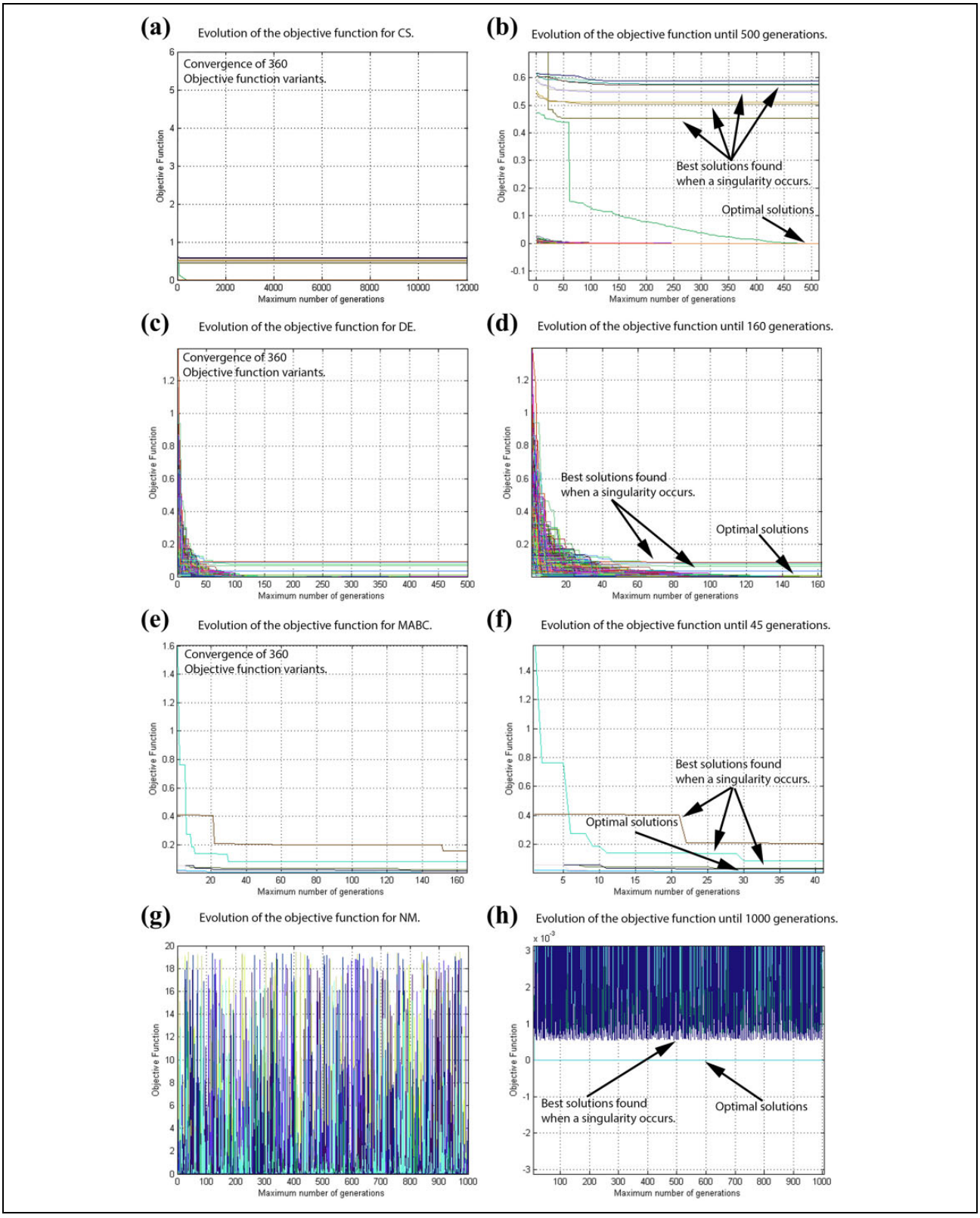

Figure 5. Objective function convergence for spiral function. Evolution of the objective function for (a) CS, (c) DE, (e) MABC, and (g) NM; (b) evolution of the objective function for CS until 500 generations.; (d) evolution of the objective function for DE until 160 generations; (f) evolution of the objective function for MABC until 45 generations; (h) evolution of the objective function for NM until 1000 generations. CS: cuckoo search; DE: differential evolution; MABC: modified artificial bee colony; NM: Newton's method.

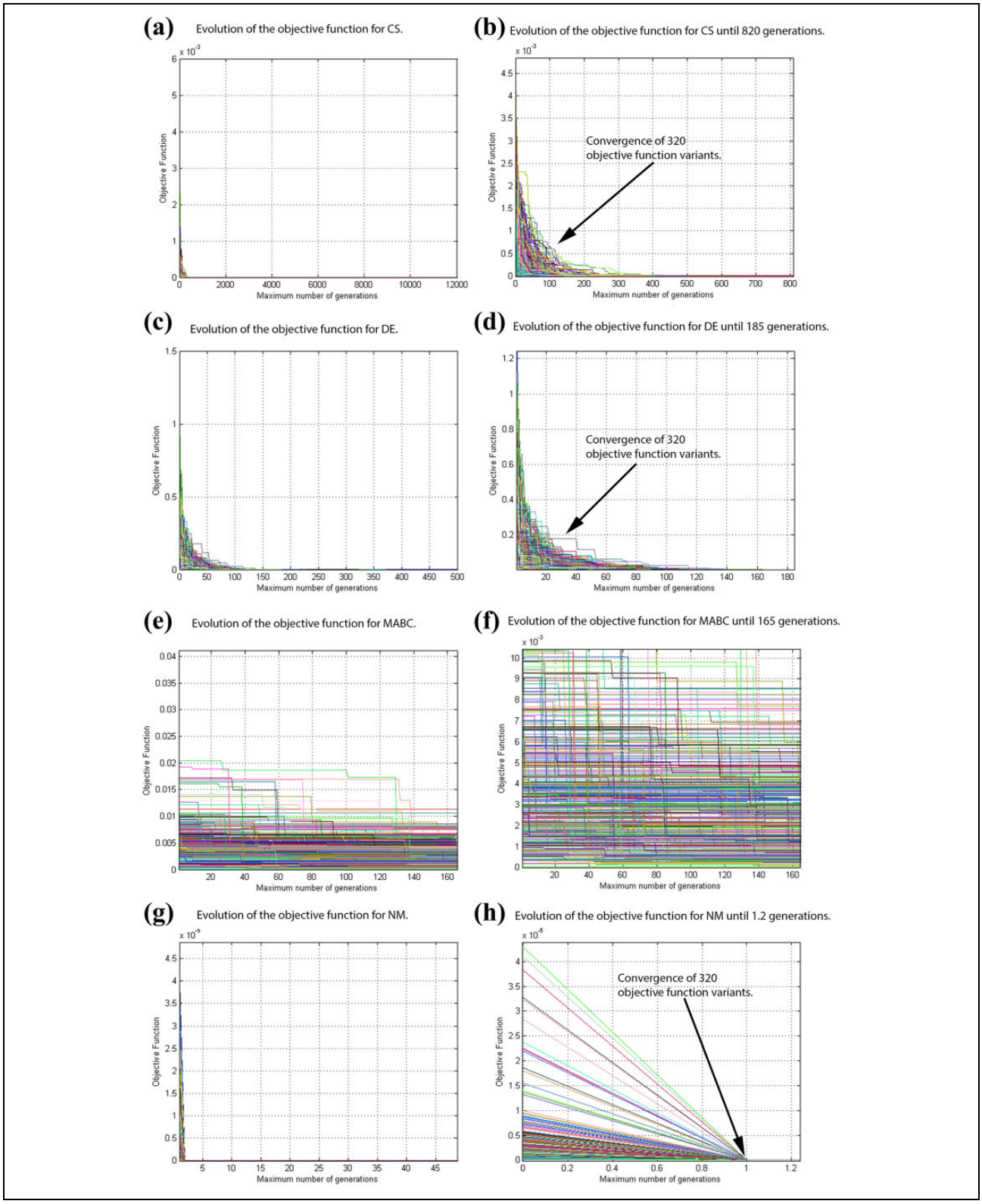

Figure 6. Objective function convergence for sinusoidal function. Evolution of the objective function for (a) CS, (c) DE, (e) MABC, (g) NM; (b) evolution of the objective function for CS until 820 generations; (d) evolution of the objective function for DE until 185 generations; (f) evolution of the objective function for MABC until 165 generations; (h) evolution of the objective function for NM until 1.2 generations. CS: cuckoo search; DE: differential evolution; MABC: modified artificial bee colony; NM: Newton's method.

Table 1. Comparison between the possible combinations analyzed from NM.

Combination	Maximum number of iterations (Max*N*)	Stop criterion value (ε)	d*x*
A	1	1e−1	0.01
B	100	1e−1	0.01
C	1000	1e−1	0.0001
D	1000	1e−1	0.01
E	1000	1e−1	0.1
F	1000	1e1	0.01
G	1000	1e−12	0.01
H	1000	1e−24	0.01

NM: Newton's method.

Table 2. Results obtained from the possible combinations for NM.

Combination	Sinusoidal trajectory	Mean OF during 60 runs	Spiral trajectory	Mean OF during 60 runs
A	Max*N* 1sc-1d*x*.01	2.9695E−06	Max*N* 1sc-1d*x*.01	0.0747
B	Max*N* 100sc-1d*x*.01	2.9695E−06	Max*N* 100sc-1d*x*.01	0.014
C	Max*N* 1000sc-1d*x*.0001	3.0845E−06	Max*N* 1000sc-1d*x*.0001	0.0223
D	Max*N* 1000sc-1d*x*.01	2.9695E−06	Max*N* 1000sc-1d*x*.01	0.0322
E	Max*N* 1000sc-1d*x*.1	0.00000224	Max*N* 1000sc-1d*x*.1	0.0223
F	SCe1 Max*N* 1000D*X*.01	2.9695E−06	SCe1 Max*N* 1000D*X*.01	0.0747
G	SCe-12 Max*N* 1000D*X*.01	4.9635E−30	SCe-12 Max*N* 1000D*X*.01	0.0226
H	SCe-24 Max*N* 1000D*X*.01	2.3156E−31	SCe-24 Max*N* 1000D*X*.01	0.0568

NM: Newton's method.

probability of abandoning a nest is equal to $P_a = 0.25$, when applying CS for most optimization problems. Starting from this recommendation, the best tuning for both trajectories was: $n = 40$, $P_a = 0.25$, and GenMax = 12,025. In Figures 5(a) and 6(a) is possible to appreciate the objective function convergence to zero, while Figures 5(b) and 6(b) show the behavior until 500 and 800 generations.

According to Karaboga,[45] for the MABC tuning implementation, the limit of trials for abandoning a source is limit = SN × D, where D corresponds to the number of design variables and SN to the number of source or food solutions. The best parameters found in addition to this criterion are: SN = 40 and GenMax = 167 for the spiral trajectory, and SN = 40 and GenMax = 165 in the case of the sinusoidal path. However, MABC was not able to find good solutions for the proposed paths, as can be seen in Figure 5(e) and (f), and Figure 6(e) and (f).

Finally, in order to find the best tuning for the NM implementation, the eight possible combinations in Table 1 were evaluated for 60 runs each, modifying the following parameters: maximum number of iterations Max*N*, stop criterion value ε, and design variable increment d*x*. The results in Table 2 were obtained from applying the eight combinations to both case studies, the sinusoidal and spiral trajectories $M_{\sin}$ and M_{spiral}, showing that the combination G is almost as good as combination H for the sinusoidal function, and at least two times better for the spiral function.

As can be seen from the results, none of the combinations for spiral trajectory was satisfactory, since this path presents two regions of singularities which cause discontinuities and elbow-configuration changes, and NM is only able to find one solution at the time as shown in Figure 5(g) and (h). In the case of the sinusoidal trajectory, all the combinations were successfully executed, presenting continuity and maintaining the elbow-configurability. For the case of the sinusoidal function, the eight possible combinations were able to follow the trajectory since it is out of singularities (see Figure 6(g) and (h)).

Table 3 includes the average values of the objective function for both trajectories using CS, DE, MABC, and NM. Every recorded pose that conforms $M_{\sin}$ and M_{spiral} involves the definition of a different OF. In general, NM, CS, and DE offer a similar average of the objective function corresponding to the sinusoidal trajectory, while in the case of the spiral trajectory, MABC does not really demonstrate superiority in comparison to NM, CS, and DE, since the mean error shown in Table 3 only evaluates how near from the optimal solution the algorithm is capable to reach (the closer the value to zero the better the solution for finding the optimal). In Figures 7(g) and 8(g), it can be appreciated from the robotic arm tracking that MABC do not offer an alternative solution in comparison with DE and CS, as can be seen in Figure 7(a) and (d), Figure 8(a) and (d).

The following experiment, reported in Table 4, consists of 60 runs of every metaheuristic algorithm to assess the quality of the obtained solutions. The run effectiveness is based on the quality in which the robot traced the path without undesired movements such as changes in the

Table 3. Statistical comparison between CS, DE, MABC, and NM after 60 runs.

			Mean of all paths	Best path	Worst path				Mean of all paths	Best path	Worst path
Sinusoidal	NM	Best	0.00E+00	0.00E+00	0.00E+00	Spiral	NM	Best	3.44E−28	3.44E−28	3.44E−28
		Mean	4.98E−30	4.98E−30	4.98E−30			Mean	0.022586	0.022586	0.022586
		Worst	3.68E−29	3.68E−29	3.68E−29			Worst	7.0233	7.0233	7.0233
		Standard deviation	7.74E−30	7.74E−30	7.74E−30			Standard deviation	0.37158	0.37158	0.37158
		Variance	5.99E−59	5.99E−59	5.99E−59			Variance	0.13807	0.13807	0.13807
		Median	8.41E−31	8.41E−31	8.41E−31			Median	3.51E−24	3.51E−24	3.51E−24
	CS	Best	8.97E−08	2.17E−09	8.81E−10		CS	Best	4.80E−08	2.19E−09	2.34E−09
		Mean	1.64E−07	1.50E−07	1.83E−07			Mean	0.002053	0.0010127	0.020034
		Worst	2.48E−07	9.20E−07	6.97E−07			Worst	0.20389	0.090399	0.58857
		Standard deviation	3.30E−08	1.20E−07	1.31E−07			Standard deviation	0.015636	0.0086366	0.093105
		Variance	1.09E−15	1.44E−14	1.70E−14			Variance	0.0002445	7.46E−05	0.0086686
		Median	1.66E−07	1.19E−07	1.52E−07			Median	1.18E−07	9.06E−08	1.12E−07
	DE	Best	1.03E−31	1.03E−31	1.03E−31		DE	Best	1.40E−31	1.40E−31	1.40E−31
		Mean	5.15E−07	1.79E−31	1.09E−05			Mean	0.0010126	0.0010126	0.0010126
		Worst	6.52E−05	2.32E−30	0.0039096			Worst	0.090399	0.090399	0.090399
		Standard deviation	4.15E−06	2.55E−31	0.0002061			Standard deviation	0.0086367	0.0086367	0.0086367
		Variance	1.72E−11	6.52E−62	4.25E−08			Variance	7.46E−05	7.46E−05	7.46E−05
		Median	1.41E−26	9.20E−32	1.66E−31			Median	1.70E−27	1.39E−31	1.76E−31
	MABC	Best	1.28E−08	1.28E−08	1.28E−08		MABC	Best	1.90E−05	2.40E−06	5.30E−05
		Mean	0.0027388	0.0024348	0.0031616			Mean	0.0009208	0.0007803	0.0013431
		Worst	0.0054443	0.013692	0.017639			Worst	0.15909	0.15938	0.15155
		Standard deviation	0.0009893	0.0022813	0.0026069			Standard deviation	0.0096486	0.0094357	0.0092653
		Variance	9.79E−07	5.20E−06	6.80E−06			Variance	9.31E−05	8.90E−05	8.58E−05
		Median	0.0026815	0.0017354	0.0024491			Median	5.09E−05	3.20E−06	0.0005074

CS: cuckoo search; DE: differential evolution; MABC: modified artificial bee colony; NM: Newton's method.

elbow-configuration sequence, and in the trajectory smoothness. The number of function evaluations (NFE) indicates that CS and DE have a similar performance, since they also reach a similar solution. The proportionality constant is $k = 0.2$ (the lower of the value, the higher the reorientation for $[0 \leq k \leq 1]$). The results show a slight improvement in the solutions found concerning CS over DE, taking into consideration the path quality, maximum number of generations GenMax, elbow-configuration sequence, and population size *Np*; this result is also shown in Figure 7(a) and (b), Figure 8(a) and (b). CS and DE can avoid the singularity by proposing a new path that fits almost completely with the original trajectory. MABC was the implemented metaheuristic with the worst performance (see Figure 7(g) to (i) and Figure 8(g)to (i)), since it could not optimize any path in the same manner that CS and DE (see Figure 7(a) and (d), Figure 8(a) and (d)), while there is an optimal solution during the trajectory, NM can converge faster than the metaheuristics mentioned above. In Figures 5(h) and 6(h) can be appreciated that the objective function takes just one iteration to converge to zero whenever there is no singularity.

Figures 7 and 8(a, d, g, and j) show the alternative trajectories generated by the robot arm with the CS, DE, MABC, and NM algorithms, using the closer elbow up and down configuration to find the best solution. Figures 7 and 8(b, e, h, and k) represent the robot arm tracking and how orientation and position were gradually varying to reach the continuous path. In Figures 7 and 8 (c, f, i, and l), the best, mean, and worst solutions obtained for the objective function can be seen. This means that every metaheuristic solution conforming the path of $M_{\sin}$ and M_{spiral} corresponds to the lines plotted in Figures 5 and 6. According to Table 4, DE had a higher number of successful runs over CS in the spiral function (96.66% vs. 80%). In the sinusoidal function, CS obtained 100% of runs successfully over 93.33% from DE, with a similar trajectory quality, as can be seen in Figure 7(a) and (d), and Figure 8(a) and (d). In the case of MABC for the sinusoidal trajectory, the algorithm improves in comparison with the spiral trajectory but not enough to optimize the path smoothly (see Figures 7(g) and 8(g)). From Figure 7(j), it can be inferred that NM is more susceptible of losing its position and elbow-configuration when orientation is not attainable, while in the case of the sinusoidal trajectory, the algorithm performs with a better quality than DE and CS (see Figure 8(k)). In general, CS and DE represent continuous sequences in the elbow-configuration for the case studies 1 and 2. For the

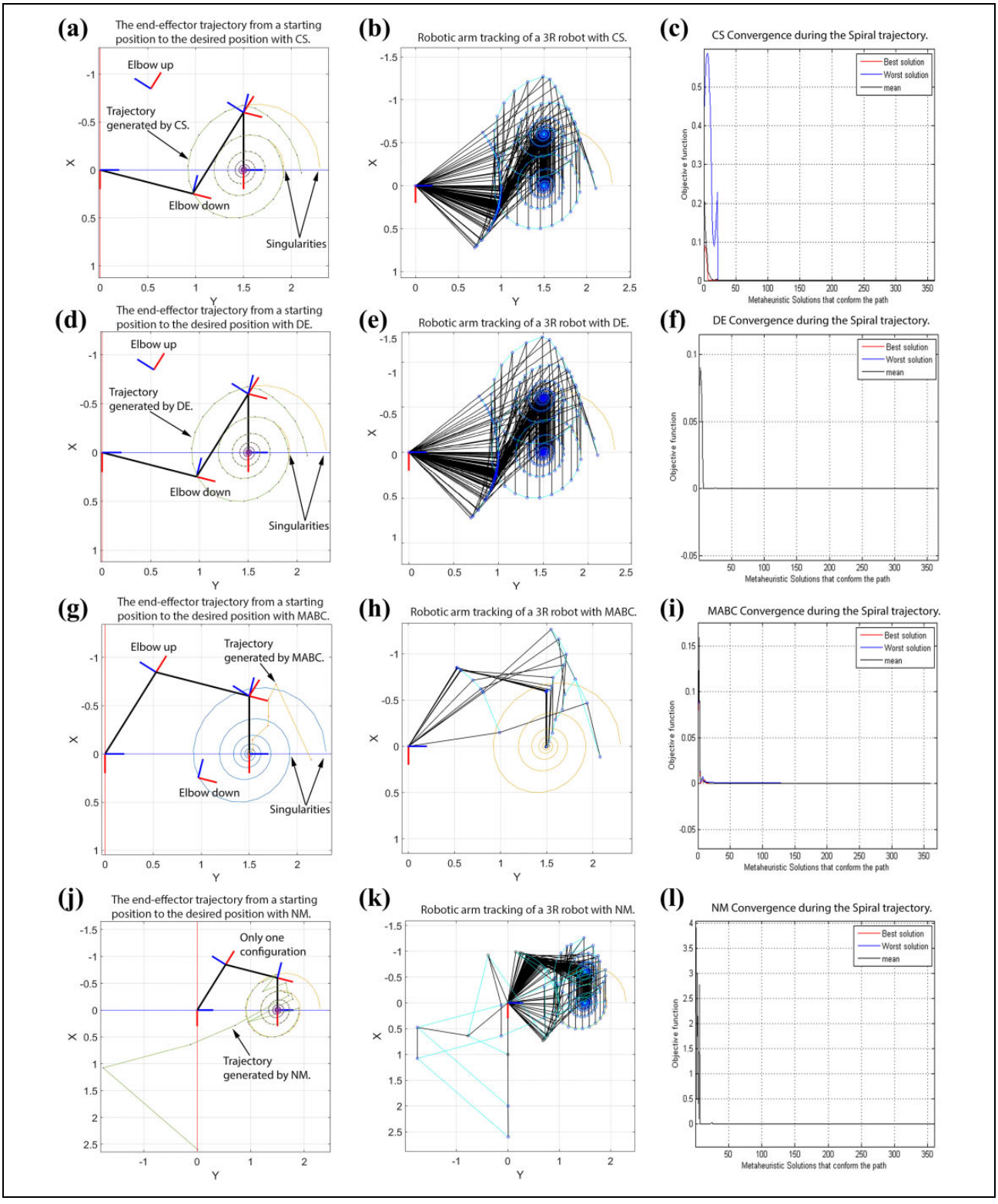

Figure 7. Graphs with spiral function implemented for CS, DE, MABC, and NM. The end-effector trajectory from a starting position to the desired position with (a) CS, (d) DE, (g) MABC, (j) NM; robotic arm tracking of a 3R robot with (b) CS, (e) DE, (h) MABC, (k) NM; (c) CS, (f) DE, (i) MABC, (l) NM convergence during spiral trajectory. (a, d, g, and j) EE path; (b, e, h, and k) robotic arm tracking; (c, f, i, and l) Cartesian error position. CS: cuckoo search; DE: differential evolution; MABC: modified artificial bee colony; NM: Newton's method; EE: end-effector; 3R: three-revolute.

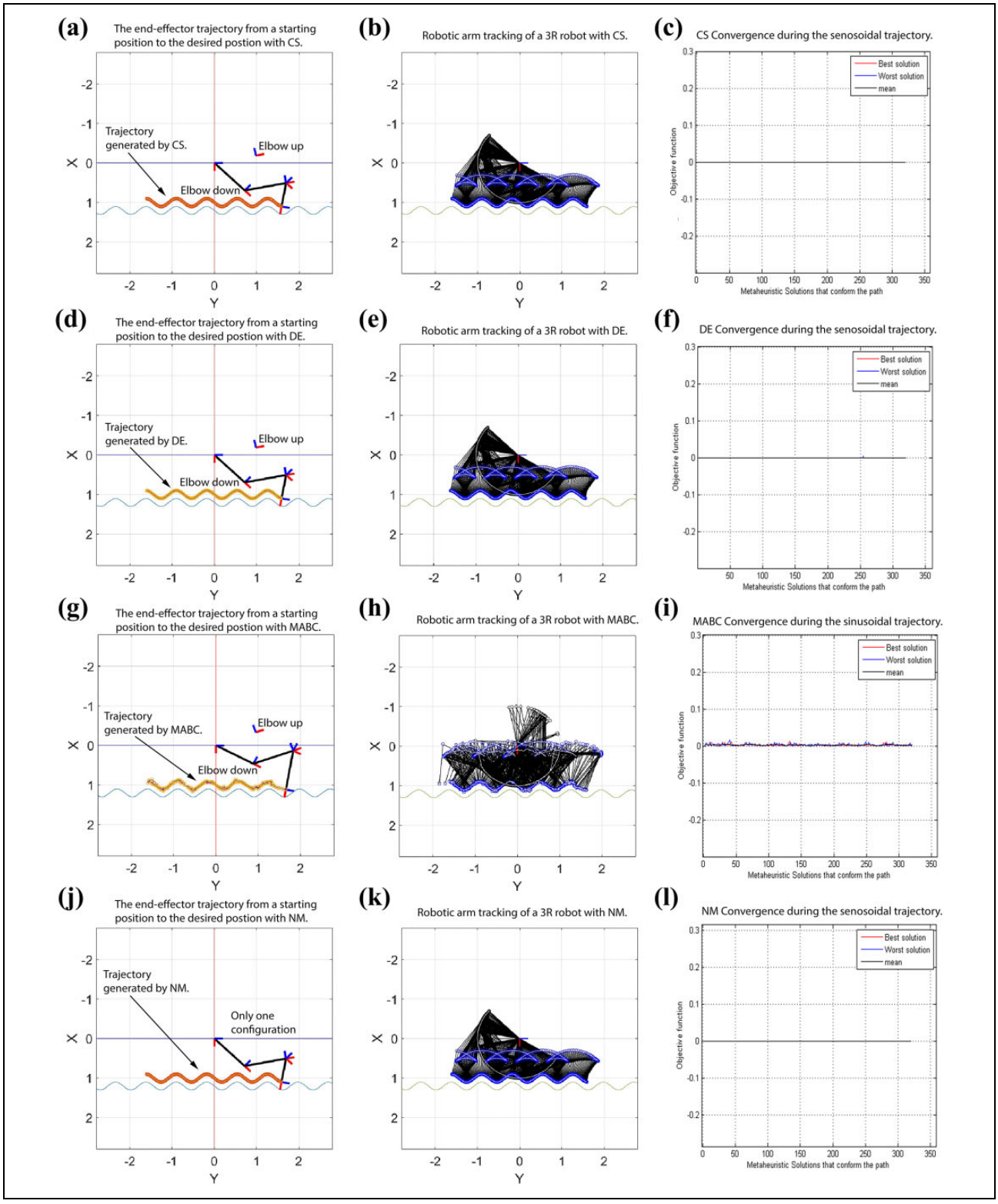

Figure 8. Graphs with sinusoidal function implemented for CS, DE, MABC, and NM. The end-effector trajectory from a starting position to the desired position with (a) CS, (d) DE, (g) MABC, (j) NM; robotic arm tracking of a 3R robot with (b) CS, (e) DE, (h) MABC, (k) NM; (c) CS, (f) DE, (i) MABC, (l) NM convergence during sinusoidal trajectory. (a, d, g, and j) EE path; (b, e, h, and k) robotic arm tracking; (c, f, i, and l) Cartesian error position. CS: cuckoo search; DE: differential evolution; MABC: modified artificial bee colony; NM: Newton's method; EE: end-effector; 3R: three-revolute.

Table 4. Comparison between DE, CS, MABC, and NM, with respect to the case studies.

Function	Algorithm	NFE	Np	GenMax	Trajectory quality	% Run effectiveness	Number of points on trajectory	Range
Spiral	DE-func1	5,410,800	30	500	V+	96.66	360	[−360 360]
	CS-func1	5,411,289	40	12,025	V−	80	360	[−360 360]
	MABC-func1	5,411,461	40	167	V/x−	—	360	[−360 360]
	NM-func1	3241	—	—	V/x−	—	360	Indef
Sinusoidal	DE-func2	4,809,600	30	500	V+	93.33	320	[−460 460]
	CS-func2	4,810,820	40	12,025	V+	100	320	[−460 460]
	MABC-func2	4,831,511	40	165	V/x−	—	320	[−460 460]
	NM-func2	41,365	—	—	V−	—	320	Indef

CS: cuckoo search; DE: differential evolution; MABC: modified artificial bee colony; NM: Newton's method.

metaheuristics, the design variables were bounded to $\pm 360°$ and $\pm 460°$ (see Table 4), reducing the search space in the proposed solutions. In NM, angular intervals are unbounded, which means that the solution of every design variable needs to be adjusted before implementation. Unlike CS and DE, NM presents path discontinuities and elbow-configuration changes at the beginning and in the middle of the spiral trajectory when experiencing singularities (see Figure 7(k)). In the case of CS, two peaks appear in Figure 7(c) for the worst solution found. This corresponds to each of the singularities in the spiral function. Since CS is not able to get an objective function convergence to zero, it is in charge of choosing the best possible solution by a greedy selection. MABC was unable to optimize the desired trajectories in Figures 7(g) and 8(g); consequently, it shows discontinuity in the trajectory and elbow configurability changes in Figures 7(h) and 8(h). Unlike CS and DE, in MABC it was impossible to find an acceptable solution to both problems even when doubling the population size and the NFE.

Conclusions

In this work, a method for avoiding singularities in trajectories of a robotic arm is developed. It consists in calculating an alternative path by solving a weighted optimization problem with a metaheuristic algorithm. For the proposed case studies, CS and DE offered greater reliability to find an alternative solution in comparison with MABC. The best numerical solution (DE) demonstrated an effectiveness of 96.66% in the spiral function and 93.33% in the sine wave function, while the second quality solution (CS) had 80% of effectiveness in the first function and 100% in the second. MABC and NM could not reach at least one successful run in the spiral trajectory. On the one hand, global search methods require knowing the multiple configurations concerning the desired pose, which makes the programming more complex. On the other hand, the bounds of every design variable reduce the solution space considerably. In general, CS and DE have a similar performance by considering a maximum number of generations and population size. Although DE seems to be more robust due to its error repeatability, CS has advantages over DE corresponding to end-effector angular displacement. Classical numerical methodologies such as NM present greater simplicity of programming and regularly converge rapidly. However, they lose their elbow-configuration when a singularity occurs, and they only contemplate the closest solution to the analysis point.

Acknowledgements

All authors thank Instituto Politécnico Nacional (COFAA, SIP, ESIME UA, and CIDETEC), and CONACYT, for all their support provided during the elaboration of this paper, and Miguel Angel Funes wants to thank CONACYT for the scholarship grant during his doctorate.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study was financially supported by Instituto Politécnico Nacional-COFAA.

ORCID iD

Miguel Angel Funes Lora http://orcid.org/0000-0002-8259-4396

References

1. Siciliano B. *Robotics modelling, planning and control*. London: Springer, 2009.
2. Nanda S. A novel application of artificial neural network for the solution of inverse kinematics controls of robotic manipulators. *Int J Intel Sys and Appl* 2012; 4: 81–91. DOI: 10.5815/ijisa.2012.09.11.
3. Anh HPH and Nguyen TN. Novel adaptive forward neural MIMO NARX model for the identification of industrial 3-DOF robot arm kinematics. *Int J Adv Robot Syst* 2012; 9(4): 104. DOI:10.5772/51277.

4. Huang Q and Liqing N. A fast convergence efficiency method of inverse kinematics for robot manipulators. *J Appl Sci* 2013; 13: 5174–5179. DOI: 10.3923/jas.2013.5174.5179.
5. Yin F, Wang Y, and Wei S. A novel hybrid electromagnetism-like algorithm for solving the inverse kinematics of robot. *Ind Robot Int J* 2011; 38: 429–440. DOI: 10.1108/01439911111132111.
6. Karpińska J, Tchoń K, and Janiak M. Approximation of Jacobian inverse kinematics algorithms: differential geometric vs. variational approach. *J Intell Robot Syst* 2012; 68(3–4): 211–224. DOI: 10.1007/s10846-012-9679-4.
7. Ratajczak J. Design of inverse kinematics algorithms: extended Jacobian approximation of the dynamically consistent Jacobian inverse. *Arch Contr Sci* 2015; 25: 35–50. DOI: 10.1515/acsc-2015-0003
8. Gouasmi M, Ouali G, and Brahim F. Robot kinematics, using dual quaternions. *Int J Robot Autom* 2012; 1: 13–30. DOI: 10.11591/ijra.v1i1.275.
9. Srisuk P, Sento A, and Kitjaidure Y. Forward kinematic-like neural network for solving the 3D reaching inverse kinematics problems. In: *14th International conference on electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON)*, Phuket, Thailand, 27–30 June 2017, pp. 214–217. IEEE. DOI: 10.1109/ECTICon.2017.8096211.
10. Pozna CR, Horváth E, and Hollósi J. The inverse kinematics problem, a heuristical approach. In: *IEEE 14th international symposium on applied machine intelligence and informatics (SAMI)*, Herlany, Slovakia, 21–23 January 2016, pp. 299–304. IEEE. DOI: 10.1109/SAMI.2016.7423024.
11. Cejka J and Cernohorsky J. Inverse kinematic problem solved by new heuristic algorithm length relaxation method. In: *18th International carpathian control conference (ICCC)*, Sinaia, Romania, 28–31 May 2017, pp. 507–511. IEEE. DOI: 10.1109/CarpathianCC.2017.7970453.
12. Calva-Yáñez MB, Niño-Suarez PA, Portilla F, et al. Reconfigurable mechanical system design for tracking an ankle trajectory using an evolutionary optimization algorithm. *IEEE Access* 2017; 5: 5480–5493. DOI: 10.1109/ACCESS.2017.2692681.
13. Zhang X, Zhang X, Yuen SY, et al. An improved artificial bee colony algorithm for optimal design of electromagnetic devices. *IEEE Trans Magnet* 2013; 49(8): 4811–4816. DOI: 10.1109/TMAG.2013.2241447.
14. Lu EH-C, Ya-Wen Y, and Su ZL-T. Ant colony optimization solutions for logistic route planning with pick-up and delivery. In: *IEEE international conference on systems, man, and cybernetics (SMC) Budapest*, Hungary, 9 February 2017, pp. 808–813. IEEE. DOI: 10.1109/SMC.2016.7844340.
15. Fernandez SA, Juan AA, Adrián JDA, et al. Metaheuristics in telecommunication systems: network design, routing, and allocation problems. *IEEE Syst J* 2018; PP: 1–10. DOI: 10.1109/JSYST.2017.2788053.
16. Asada H. *Robot analysis and control*. Hoboken: Wiley, 1986.
17. Tsai LW. *Robot analysis: the mechanics of serial and parallel manipulators*. Hoboken: Wiley, 1999.
18. Craig JJ. *Introduction to robotics: mechanics and control*. Upper Saddle River: Pearson/Prentice Hall, 2005.
19. Ceccarelli M. *Fundamentals of mechanics of robotic manipulation*. Netherlands: Springer, 2013.
20. Rao SS. *Engineering optimization: theory and practice*. Hoboken: Wiley, 1996.
21. Papalambros P and Wilde DJ. *Principles of optimal design: modeling and computation*. Cambridge, New York: Cambridge University Press, 2000, p. 390.
22. Takehiko O and Kanada H. Solution for ill-posed inverse kinematics of robot arm by network inversion. *Journal of Robotics* 2010; 2010: 9. DOI: 10.1155/2010/870923.
23. Bazaraa MS. *Nonlinear programming: theory and algorithms*. Hoboken: Wiley; 2013.
24. Hromkovič J. *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics*. Heidelberg: Springer, 2013.
25. Chakraborty UK. *Advances in differential evolution*. Heidelberg: Springer, 2008.
26. Gu W. Genetic algorithms with stochastic ranking for optimal channel assignment in mobile communications. *Computational and Information Science*. Berlin, Heidelberg: Springer, 2005, pp. 154–159.
27. Dominguez-Isidro S, Mezura-Montes E, and Leguizamon G. Memetic differential evolution for constrained numerical optimization problems. In: *IEEE congress on evolutionary computation*, Cancun, Mexico, 15 July 2013, pp. 2996–3003. IEEE. DOI: 10.1109/CEC.2013.6557934.
28. Domınguez-Isidro S, Mezura-Montes E, and Leguizamon G. Performance comparison of local search operators in differential evolution for constrained numerical optimization problems. In: *IEEE symposium on differential evolution (SDE)*, Orlando, FL, USA, 9–12 December 2014, pp. 1–8. IEEE. DOI: 10.1109/SDE.2014.7031530.
29. Aguilar-Justo AE, Mezura-Montes E, and Coello CAC. Memetic modified artificial bee colony for constrained optimization. In: *2014 IEEE international autumn meeting on power, electronics and computing (*ROPEC*)*, 5–7 Nov 2014, 2014, pp. 1–6. DOI: 10.1109/ROPEC.2014.7036348.
30. Panigrahi BK, Das S, Suganthan PN, et al. Swarm, evolutionary, and memetic computing. In: *1st International conference on swarm, evolutionary, and memetic computing, SEMCCO 2010*, Chennai, India, 16–18 December 2010. Berlin, Heidelberg: Springer.
31. Hart WE. *Recent advances in memetic algorithms*. Heidelberg: Springer, 2006.
32. Mezura-Montes E, Reyes-Sierra M, and Coello CAC. Multi-objective optimization using differential evolution: a survey of the state-of-the-art. In: *Advances in differential evolution* (ed UK Chakraborty)*, 2008*, Vol. 143, pp. 173–196. Berlin, Heidelberg: Springer. DOI:10.1007/978-3-540-68830-3_7.
33. Feoktistov V. *Differential evolution: in search of solutions*. New York: Springer, 2007.
34. Zhang J. *Adaptive differential evolution: a robust approach to multimodal problem optimization*. Heidelberg: Springer, 2009.

35. Antunes C, GP Liu, Yang JB, et al. Multiobjective optimisation and control, series: engineering systems modelling and control, research studies press (2003). *European Journal of Operational Research* 2006; 174: 668–671. DOI: 10.1016/j.ejor.2005.04.003.
36. Mezura-Montes E and Cetina-Domínguez O. Empirical analysis of a modified artificial bee colony for constrained numerical optimization. *Applied Mathematics and Computation* 2012; 218: 10943–10973. DOI: 10.1016/j.amc.2012.04.057.
37. Qing A. *Differential evolution: fundamentals and applications in electrical engineering*. Hoboken: Wiley, 2009.
38. Onwubolu GC. *New optimization techniques in engineering*. Heidelberg: Springer, 2013.
39. Yang XS. *Cuckoo search and firefly algorithm: theory and applications*. New York: Springer International Publishing, 2013.
40. Joshi AS, Kulkarni O, Kakandikar GM, et al. Cuckoo search optimization: a review. *Mater Today Proc* 2017; 4(8): 7262–7269. DOI: 10.1016/j.matpr.2017.07.055.
41. Yang XS. *Nature-inspired metaheuristic algorithms*. UK: Luniver Press, 2010.
42. Karaboga D and Basturk B. Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. In: *Foundations of Fuzzy Logic and Soft Computing*, 2007, pp. 789–798. Berlin, Heidelberg: Springer. DOI: 10.1007/978-3-540-72950-1_77.
43. Abu-Mouti FS and El-Hawary ME. Overview of artificial bee colony (ABC) algorithm and its applications. In: *IEEE international systems conference SysCon 2012*, vancouver, BC, Canada, 19–22 March 2012, pp. 1–6. IEEE. DOI: 10.1109/SysCon.2012.6189539.
44. Nakamura S. *Análisis numérico y visualización gráfica con MATLAB*. Upper Saddle River: Prentice-Hall Hispanoamericana, 1997.
45. Karaboga D and Akay B. A modified artificial bee colony (ABC) algorithm for constrained optimization problems. *Appl Soft Comput* 2011; 11: 3021–3031. DOI: 10.1016/j.asoc.2010.12.001.