International Journal of Combinatorial Optimization Problems and Informatics, 15(1), Jan-April 2024, 7-19. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2024.v15i1.375

www.editada.org

Evolutionary computation, an alternative solution to nonlinear optimization problems

Francisco J. Espinosa-Garcia[1], Ricardo Tapia-Herrera[2], Tonatiuh Cortés-Hernández[3], and Jesús A. Meda-Campaña[4]*

[1] Laboratory of Robotics and Mechatronics (LARM 2), University of Rome Tor Vergata, Rome, Italy.
[2] CONAHCYT-SEPI-ESIME Zacatenco, Instituto Politécnico Nacional, Av. IPN, Col. Lindavista, Ciudad de México, México.
[3] Departamento de Mecatrónica, Universidad Politécnica de Pachuca, 43830, Zempoala, Hidalgo, México.
[4] SEPI-ESIME Zacatenco, Instituto Politécnico Nacional, Av. IPN, Col. Lindavista, Ciudad de México, México.
E-mails: fjeg_1234@hotmail.com, tonatiuh@upp.edu.mx, jmedac@ipn.mx
*Corresponding author: rtapiah@ipn.mx

Abstract. Analytical and numerical methods have been applied to solve problems in engineering. However, in some practical cases, they usually fail when there is a certain degree of complexity, for instance, when there is a certain lack of information about the elements of the system and when the unknowns are functions. These types of problems are often called nonlinear optimization problems. As an alternative to solving them, evolutionary computation methods are usually implemented, although they do not generate an exact solution, and provide a series of approximations that are generally feasible. In this context, the objective of this work is to briefly highlight the most typical characteristics of these type of algorithms, some advantages, and the importance of its use today. Due to the wide variety of existing methods, it would become complex to explain all of them in detail, so only a description of the differential evolution (DE) algorithm will be made because it is one of the most used and because there is current research that seeks to improve its performance.

Keywords: Crossover, Fitness Function, Mutation, Population, Selection, Differential Evolution

Article Info
Received June 28, 2023
Accepted Dec 11, 2023

1 Introduction

Currently, optimization is a useful tool between researchers and engineers in different areas of science, like the development of processes, design of elements, machines, and tools where it has been improved aspects such as a maximum use of the resources, shorter development of times, and guarantee that the generated product mostly covers the needs (Padmanabhan et al., 2011; Zolpakar, Yasak & Pathak, 2021; Zolpakar et al., 2020; Shaikh, Jain & Pathak, 2016; Fountas & Vaxevanidis, 2020), so that its performance is adequate based on the application for which it was designed. In this way, it is ensured that the presence of errors or defects is almost zero compared to developments based on the trial-and-error method.

One of the main attributes of optimization is the ability to describe a real problem in mathematical terms, which results in the formulation of functions that allow the problem to be visualized in a formal way and treated in a rigorous manner (Padmanabhan et al., 2011). In the past, the most popular methods implemented to solve non-linear optimization problems were of the analytical and mathematical type, generally known as "descent" (Zolpakar, Yasak & Pathak, 2021). These are classified into the first-order ones, which include the gradient, conjugate gradient, and Fletcher-Reeves methods (Zolpakar et al., 2020; Shaikh et al., 2016). The second-order methods include Newton's methods and their variations (Fountas & Vaxevanidis, 2020). Finally, there are methods known as quasi-Newton, some methods are Davison-Fletcher-Powell (DFP) and Broyden-Fletcher-Golfarb-Shanno (BFGS) [Lav et al., 2009). The problem with the above methods is that they are limited when the problem, besides

being non-linear, presents restrictions, which in the general case could be of non-linear type. To solve this type of problem, other methods known as sequential techniques of unconstrained minimization (SUMT) are usually implemented (Tham et al., 1998). These techniques seek to formulate the problem as one without constraints, then several unrestricted problems must be solved to find the solution of one with constraints. In this category are the exterior penalty methods (Oguntola & Lorentzen, 2021), the interior penalty method (Brenner, Owens & Sung, 2008), and the extended interior penalty method (Kim & Kim, 1993). Despite presenting a good performance for solving some practical problems, these methods are usually limited when the objective function does not meet two primary requirements, being derivable and unimodal (that is, it presents a single maximum and minimum value). This is an impediment because real-life problems sometimes cannot be expressed as differentiable functions and are generally multimodal. In this context, evolutionary computation methods have a relevance since, although they cannot give an exact solution, they can generate approximate solutions with reasonable resources to this type of problem with a high degree of complexity (Vikhar, 2016).

Evolutionary computation methods are stochastic, that is, they use random-type processes to search for solutions. Due to this characteristic, it is difficult to analyze the behavior of these algorithms, so most of their properties have been discovered experimentally (Vikhar, 2016).. Its operation is generally based on the behavior of biological phenomena. Some of the most popular are Genetic Algorithms (GA) and Differential Evolution Algorithms (DE) which are based on the theory of evolution of species formulated by Charles Darwin, having selection, mutation, and growth as main operators. The particle swarm optimization (PSO) algorithm, unlike GA, is a method that is based on the behavior of insect swarms in nature, emulating the evolution in collective behavior resulting from a combination of individual decisions (Wang, Tan & Liu, 2018). The operators in this method are the velocity and the movement of the particles. Another popular algorithm is the artificial bee colony (ABC) algorithm; this algorithm is inspired by the behavior of bees in search of honey (Nozohour-leilabady & Fazelabdolabadi, 2016). The main advantages of this type of algorithm are that they do not depend on the structure of the problem, that is, they can be used for a wide range of problems so complex that they may contain simulations or experimental models. They are parallelizable algorithms (the sequential code becomes multithreaded and/or vectorized) with the aim of using multiple processors simultaneously. They can solve non-differentiable and multimodal problems. Unlike gradient-based algorithms, the function's gradient is not needed. These kinds of algorithms often incorporate some form of randomness to escape local minima. In addition, they are easy to implement.

2 Fitness function

For the implementation of an optimization process, it is important to transfer the problem to a mathematical representation that allows the inclusion of the main objectives to be achieved. This representation is generally known as the objective function, which is used to measure the quality of the solutions, either to maximize or minimize resources (Baresel, Sthamer & Schmidt, 2002). The general representation of an objective function incorporates various mathematical expressions and constraints as follows:

$$min/max\ f(x),\ \ X=(x_1, x_2,\ldots,x_i,\ldots,x_N)\ , \tag{1}$$

Subjected to:

$$g_i(X) < b_j,\ \ j=1, 2,\ldots, m$$

$$x_i^{(L)} \leq x_i \leq x_i^{(U)},\ i= 1, 2,\ldots, N.$$

Where $f(x)$ represents the objective function, X is the set of variables to consider, $g_i\ (X)$ represents the constraints of the problem, b_j the constraint constants, m the total number of restrictions; $x_i^{(L)}$ and $x_i^{(U)}$ represent the lower and upper bounds for each variable.

3 Differential Evolution Algorithm

The Differential Evolution (DE) Algorithm was proposed in 1994 by Kenneth Price and Rainer Stern. This is a very popular method since it can reach the global optimum in multimodal, non-differentiable, and non-linear functions. Its main features are that it is easy to implement, requires few adjustment parameters, and can be parallelized to handle functions with high

computational cost. The structure of the algorithm is based on perturbing the member of the population with scaled differences from the members of the same population (Mallipeddi, et al., 2011).

The stages of this algorithm are initialization of the population (in a stochastic way), differential mutation on the population, which is conditioned to the fulfillment of a criterion, crossing between individuals with the purpose of increasing diversity and finally a selection is made by means of a suitable criterion (Mallipeddi, et al., 2011). Below is a brief explanation of each of the stages mentioned. The population is randomly generated between the lower ranges $x_i^{(L)}$ and upper ranges $x_i^{(U)}$ using the following expression:

$$\begin{gathered} x_{j,i}=x_i^{(L)} + rand\ (0,1) * (x_i^{(U)} - x_i^{(L)}), \\ j = 1,2, \ldots, d, \\ i = 1,2, \ldots, NP. \end{gathered} \tag{2}$$

The mutation is an operator used to randomly alter, under a certain condition, the individuals of the previously generated population (Mallipeddi, et al., 2011). This condition comes from the generation of a random number that must be less than the parameter M whose value can be defined between 0 and 1. In this particular algorithm, the mutation takes two vectors X_{r1}, X_{r2} from the population to perform a scaled difference to a third population vector X_{r0}. The following expression shows the application of the operator to generate a vector called a mutant.

$$V_{n,i} = X_{r0} + F * (X_{r1} - X_{r2}). \tag{3}$$

The scaling factor F is a value between 0 and 1 that controls the level at which the population will evolve, if the value is close to zero it will converge faster than if it is close to 1 (Mallipeddi, et al., 2011). Also, there are some popular variations of this operator, they are listed below.

Mutation best/1 is a variation where two individuals are taken randomly from the population (X_{r1}, X_{r2}). The $X_{best,g}$ is the best global individual of the current generation. The corresponding expression is:

$$V_{n,i} = X_{best,g} + F * (X_{r1} - X_{r2}). \tag{4}$$

Mutation rand/2 unlike to Mutation best/1 implements 5 individuals taken randomly from the population (X_{r0}, X_{r1} X_{r2}, X_{r3} X_{r4}) considering that they will be different. This strategy is commonly used when the problem needs population with more diversity. The corresponding expression defined as:

$$V_{n,i} = X_{r0} + F * (X_{r1} - X_{r2}) + F * (X_{r3} - X_{r4}). \tag{5}$$

The crossover operator implies the choice of two individuals, the mutant vector $V_{n,i}$ and the original individual $X_{n,i}$ for the exchange of segments of the population in order to maintain the diversity. The crossing vector $U_{n,i}$ is calculated as:

$$U_{n,i} = \begin{cases} V_{n,i} & \text{rand}(0,1) \le CR \\ X_{n,i} & \text{otherwise} \end{cases}. \tag{6}$$

The CR parameter can have a value between 0 and 1.

Selection is used to find the best individuals (results of mutation and crossover operations) that should be copied for the next generation. The selection method is Greedy of the elitist type since it makes sure to select the best solution found (Mallipeddi, et al., 2011). In other words, $x_{n,i}$ will remain in the population until the next generation unless the test vector $U_{n,i}$ has a better fitness value (Equation 8).

$$X_{n,i+1} = \begin{cases} U_{n,i} & f\left(U_{n,i}\right) \leq f\left(X_{n,i}\right) \\ X_{n,i} & \text{otherwise} \end{cases} \tag{7}$$

The flowchart of the algorithm is shown in Figure 1.

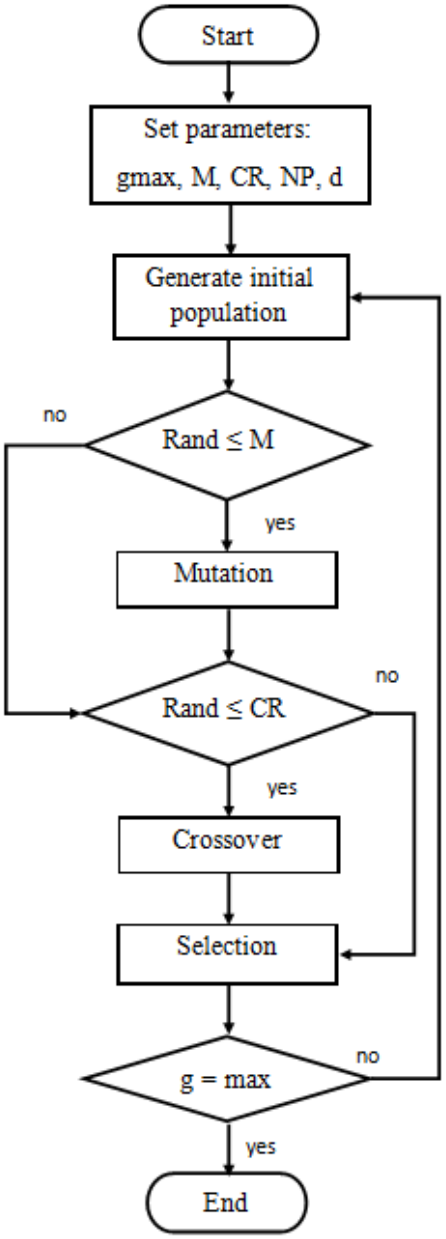

Fig. 1. Flowchart for the Differential Evolution algorithm.

4 Implementation and Results

For the implementation of the DE algorithm, two examples are used. The first is the Rastrigin function (Valdez & Melin, 2007). This function is non-convex and a typical example of a non-linear multimodal function. Finding the minimum is a difficult problem due to its large search space and many existing local minima (Figure 2).

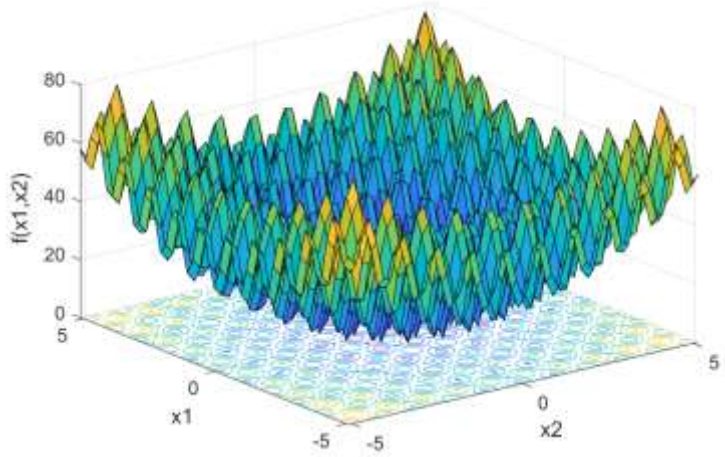

Fig. 2. Rastrigin function which presents multiple local maximal and minimal.

The problem is to minimize the function defined by:

$$f_1(x_1, x_2) = 10d + \left[x_1^2 - 10\cos\left(2\pi x_1\right)\right] + \left[x_2^2 - 10\cos\left(2\pi x_2\right)\right] \tag{8}$$

Subject to:

$$-5.12 \le x_1 \le 5.12$$
$$-5.12 \le x_2 \le 5.12$$

For implementation of the algorithm the following parameters were used: number of generations = 50, M = 0.2, CR = 0.2, F = 0.8 and the number of populations = 50. To demonstrate the heuristic characteristic and the approximation power of the algorithms, 15 experiments were carried out, 5 with classical mutation variation, 5 with Mutation best/1 and rest with Mutation rand/2. They were run on a computer with a Core i9 processor, 16 GB of RAM and a speed of 2.6GHz.

The obtained results are shown in Table 1.

Table 1. Obtained results from optimization process.

Test	Mutation variation	x_1	x_2	Fitness
1	Common variation	0.9951	1.4435e-4	0.9949
2	Common variation	4.2519e-06	0.9950	0.9950
3	Common variation	-0.02927	0.00536	0.17526
4	Common variation	-0.01623	0.019206	0.12537
5	Common variation	-0.0333e-3	0.5105e-3	0.0519e-3
6	Mutation best/1	-1.2187e-4	3.9217e-05	3.2880e-8
7	Mutation best/1	0.9951	7.3208e-06	0.9949
8	Mutation best/1	1.2089e-4	7.0949e-5	2.7799e-8
9	Mutation best/1	1.2034e-6	-0.9948	0.9949
10	Mutation best/1	-0.0016e-08	-0.2904e-08	0
11	Mutation rand/2	2.1858e-4	1.9142e-4	3.6211e-8
12	Mutation rand/2	3.9661e-5	2.6661e-4	3.2683e-7
13	Mutation rand/2	4.0018e-5	1.3885e-4	8.0563e-8
14	Mutation rand/2	-1.1672e-4	1.0008e-4	5.7736e-9
15	Mutation rand/2	2.6731e-5	-1.2218e-4	1.4167e-7

Based on the results of Table 1, the best-obtained result was $0.0519x10^{-3}$ using the common variation corresponding to iteration 5, in addition 80% of the approximates have a fitness of $1x10^{-1}$. Referring to Mutation best/1 the results are divided into 40% corresponding to $1x10^{-1}$, 40% in the range $1x10^{-8}$, and 20% with an error of 0. Using the Mutation rand/2 variation the results show that 40% of solutions are in the range of $1x10^{-7}$, 40% in the range of $1x10^{-8}$, and 20% in the range of $1x10^{-9}$. In general, considering the performance of the algorithm all the approximations obtained are acceptable (i.e., the solutions are closer to zero).

Regardless of the mutation strategy used, the algorithm yields 15 possible results, where 40% represents solution with an error of $1x10^{-1}$, 6.6% with an error of $1x10^{-3}$, 13.3% with fitness of $1x10^{-7}$, 26.6% results with $1x10^{-8}$ of error and 6.6% results in a range of $1x10^{-9}$. Fortunately, the best global result has a fitness of 0, which means that the algorithm was able to find an exact solution. The performance of the algorithm in this iteration is shown in Figure 3. As can be seen, from the beginning (i.e., from iteration 1) the fitness obtained is very close to zero (in the range of $1x10^{-15}$). Finally, the algorithm converges to zero in iteration 7. It is considered fast due that on average the best solutions obtained converge after iteration 20 and they are just approximations.

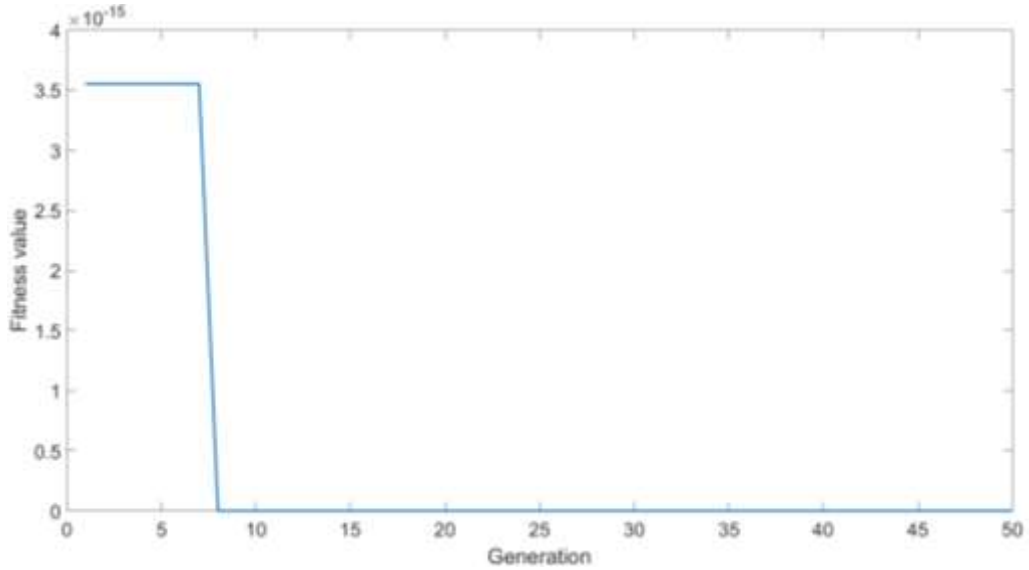

Fig. 3. Convergence of best fitness for differential evolution.

The second example is more complex. Here, the problem is to find the lengths of a mechanism implemented in a finger. The optimization problem was taken from (Espinosa-Garcia et al., 2021). Basically, the problem is achieved that the point P_d (Figure 4) follows the target points obtained from the flexion-extension movement. The values of the target points are shown in Table 2.

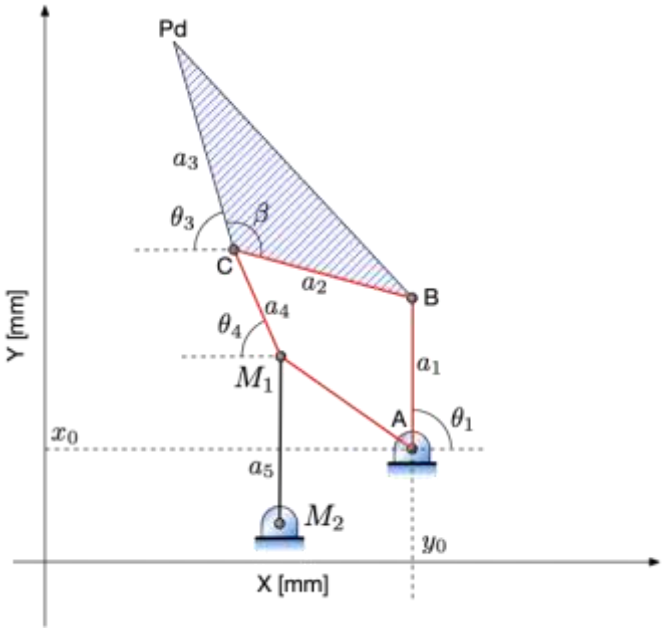

Fig. 4. Parameters representation for the proposed four-bar mechanism (Espinosa-Garcia et al., 2021).

Table 2. Desired points (Espinosa-Garcia et al., 2021).

	Pd1	Pd2	Pd3	Pd4	Pd5
x	0	-17.95	-35.92	-38.03	34.77
y	47.76	34.40	21.67	7.36	-12.43

The problem is to minimize the following expression:

$$\min \sum_{i=1}^{n} \left[\left(Pd_x^i - P_x^i \right)^2 + \left(Pd_y^i - P_y^i \right)^2 \right] + h(x) \tag{9}$$

Subjected to:

a) $\theta_1^i > \theta_1^{i+1} > \ldots > \theta_1^{i+n}$ where $i = 1$ and $n = 5$.

b) $x_i \in [Li_i, Ls_i]$ where $X = [x_1, x_2, x_3, x_4, x_5, x_6]$

where P_x^i and P_y^i represent the path generated by the mechanism. As can be seen, in Equation 9 parameter *h(x)* has been added in order to evaluate the sequence condition for the input angle (represented by θ_1^{i+1}). If the condition is true *h(x)* = 0, otherwise *h(x)* = 1. Li_i *and* Ls_i are the ranges for the design variables. Also, the corresponding equations to calculate the points throw the optimization process are shown below.

$$B_{x0} = a_1 \cos\theta_1 \tag{10}$$

$$B_{y0} = a_1 \sin\theta_1 \tag{11}$$

$$C_{x0} = B_x + a_2 \cos\theta_2 \tag{12}$$

$$C_{y0} = B_y + a_2 \sin\theta_2 \tag{13}$$

$$P_{x0} = C_x + a_3 \cos\theta_{\alpha_3+\beta} \tag{14}$$

$$P_{y0} = C_y + a_3 \sin\theta_{\alpha_3+\beta} \tag{15}$$

Where P_x^i and P_y^i represent the path generated by the mechanism. As can be seen, in Equation 9 parameters *h(x)* have been added in order to evaluate the sequence condition for the input angle (represented by θ_1^{i+1}). If the condition is true then *h(x)* = 0, otherwise *h(x)* = 1. Li_i *and* Ls_i are the ranges for the design variables. Also, the corresponding equations to calculate the points throw the optimization process are shown below:

Design variables: $\left[\theta_1^1,...,\theta_1^5,a_1,a_2,x_0,y_0,\beta\right]$

Desired points: $\left[\left(P_{xd}^1,P_{xd}^5\right),...,\left(P_{yd}^1,P_{yd}^5\right)\right]$ shown in Table 2

Limits of the variables: $a_1 \in [20,50]$, $a_2 \in [6,16]$, $x_0 \in [-5,15]$, $y_0 \in [-20,20]$, $\beta \in [-65,180]$, and $\theta \in [0,2\pi]$

The parameters used for the algorithm are:

Generation number: 300
Crossover value: 0.6
Mutation value: 0.1
Population: 100
Scale factor: 0.5

In this case 30 experiments were carried out, 10 per each mutation variation. The program was executed on a computer with a Core i9 processor, 16 GB of RAM and a speed of 2.6GHz.

The obtained results are shown in Tables 3, 4, and 5. They show the number of the test, values of each parameter and their fitness value.

Table 3. Obtained results from optimization process using rand/1 strategy.

Test	a_1	a_2	a_4	rd	x_0	y_0	Fitness
1	24.3030	14.9157	33.9725	22.4005	12.3148	5.0415	1.5822
2	15.9999	33.8358	31.6758	21.8271	1.5648	-1.0905	3.1967
3	30.5232	15.9999	42.1660	24.3338	12.0337	4.0333	5.1349
4	27.8374	7.9933	29.7973	31.4786	11.7794	-8.0003	0.4487
5	24.1767	10.1415	37.5128	32.1154	13.6280	19.3293	0.0013
6	25.8764	15.1561	28.2025	28.3689	14.3245	-8.2985	0.0419
7	45.1694	15.9996	32.0971	25.4870	2.2419	12.1034	4.0285
8	21.5547	7.9514	42.9915	34.8578	1.4869	16.0489	0.0114
9	33.0984	15.9021	37.1562	24.4691	0.3884	17.1464	0.0296
10	31.8459	13.8851	40.7737	24.3860	-3.5149	-5.5112	0.0007

Table 4. Obtained results from optimization process using best/1 strategy.

Test	a_1	a_2	a_4	rd	x_0	y_0	Fitness
1	24.6394	11.4333	23.8129	27.4862	8.9658	-13.3401	1.3154
2	23.1392	14.1194	35.8427	23.0909	14.9603	13.7722	0.0005
3	22.5589	12.7767	35.5301	24.3581	7.6679	16.9083	0.9389
4	27.2767	14.7712	41.6917	24.4053	2.3028	7.2225	0.2843
5	26.1739	9.7601	36.8427	26.9694	5.9285	11.9301	0.1257
6	28.2679	11.6685	26.5740	25.1131	0.9931	17.7730	0.0016
7	32.3392	11.2852	40.4987	27.1674	4.6405	-16.3323	0.0224
8	38.5588	15.8412	43.1319	25.3202	-2.6399	-9.4468	0.0503
9	28.3632	9.0977	32.4859	34.8396	1.0317	-7.805	0.4091
10	28.3192	15.3181	28.9841	25.5605	6.4878	16.2565	0.0000546

Table 5. Obtained results from optimization process using rand/2 strategy.

Test	a_1	a_2	a_4	rd	x_0	y_0	Fitness
1	34.1180	15.9919	40.1431	20.3649	14.2975	3.5804	0.0005
2	21.6259	10.9888	32.9151	26.8714	2.4709	19.4979	0.0088
3	28.3207	10.0225	25.6173	28.2256	-3.7477	12.5013	0.0018
4	27.8586	8.5352	42.5234	25.4989	9.4493	-13.1548	0.0005
5	35.5704	15.7778	44.4807	25.6712	3.0291	-4.4687	0.0001
6	28.7880	13.5139	39.1024	28.2531	7.2513	13.8420	0.0004
7	26.2914	9.7418	24.3337	32.9169	5.6564	-5.9788	0.0379
8	29.5902	9.0022	23.1935	26.7245	2.5706	0.9389	0.1415
9	23.4158	10.2971	30.3672	23.9450	14.4005	-9.8019	5.37471E-6
10	25.5396	10.6529	23.8024	26.8969	2.5183	-12.406	0.0001

Based on the results of Table 3 corresponding to the rand/1 strategy, the results show that 40% of the solutions are unfeasible (tests 1, 2, 3, and 7) and just 60% feasible. From the 60%, 10% of the solutions are in the range of $1x10^{-1}$ (test 4), 30% has a fitness value with error of $1x10^{-2}$, 10% in order to $10x10^{-3}$ and other 10% corresponds to the error in range of $1x10^{-4}$. The best result was obtained in test 10. Figure 4 shows the convergence of the test. The initial value is high (1600), after iteration 5 the fitness value is 1000 after that, the value continues decreases to zero. Final value is obtained in generation 48 (with a value of $7x10^{-4}$).

From Table 4, the unfeasible results represent 10%. Referring to the feasible solutions, 40% are in the range of $1x10^{-1}$, 20% in the range of $1x10^{-2}$, 10% 1x10-3, 10% 1x10-4 and 10% $1x10^{-5}$. The convergence of the best fitness obtained in test 10 is shown in Figure 5. The same as the fitness of Figure 4 the initial value is high (1580). The difference with respect to Figure 4, the algorithm approaches to zero faster because it achieves the best value in generation 40. Moreover, the test is stopped in generation 240 instead of 300 because the algorithm cannot improve the value of the fitness.

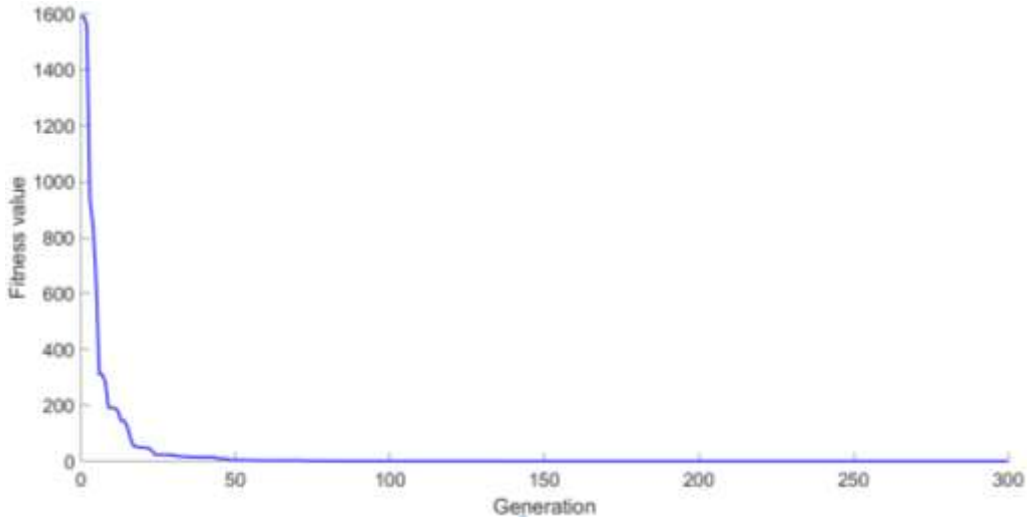

Fig. 4. Convergence of best fitness rand/1 strategy.

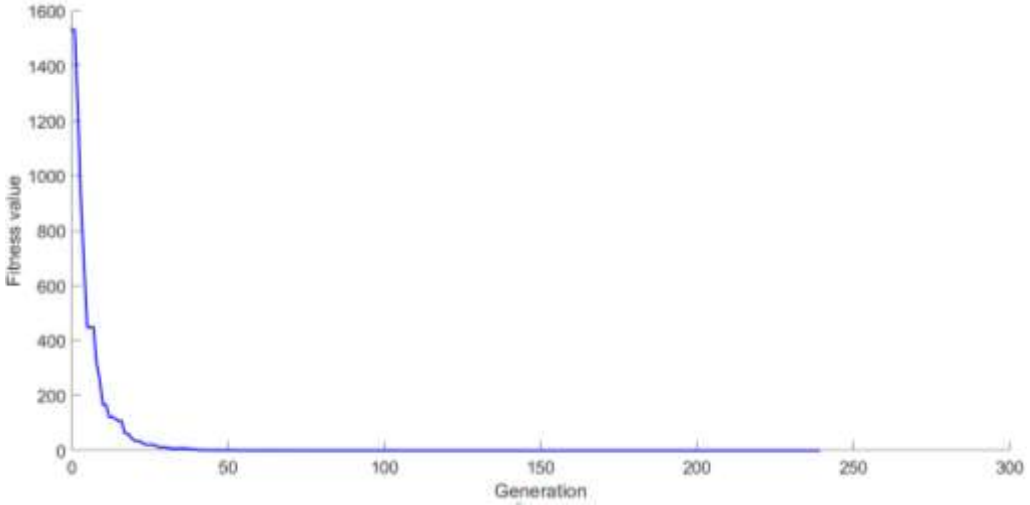

Fig. 5. Convergence of best fitness best/1 strategy.

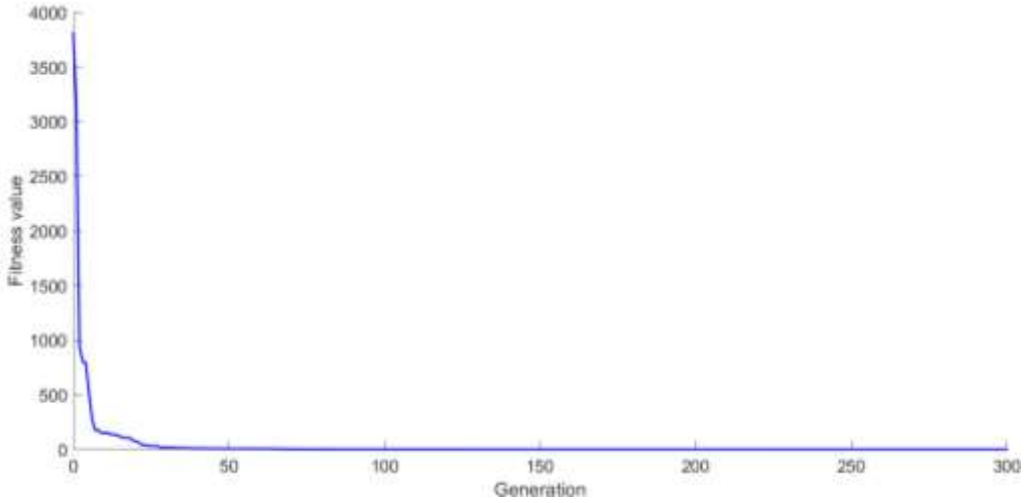

Fig. 6. Convergence of best fitness rand/2 strategy.

The results of Table 5 show 100% of feasible solutions, the quality of this can be described as follows: 10% in range of $1x10^{-1}$, 10% corresponds to range $1x10^{-2}$, 20% in range of $1x10^{-3}$, 50% with value of $1x10^{-4}$ and 10% in range of $1x10^{-6}$. Unlike the other cases, the solutions presented are closer to zero. In this case the best fitness was found in test 9. The convergence graph is shown in Figure 6. Unlike the other cases, the initial fitness value is high (4000). But the convergence is faster than others because the fitness value is closer to zero in generation 26. In terms of percentage, the convergence of this obtained best is 41.66% faster than the best of Table 1, and 50% faster than the best of Table 2.

In order to get a better view of the results obtained, Figure 7 shows the plot of the trajectories corresponding to the desired trajectory, the trajectory obtained in the original work, and the trajectories obtained using the DE algorithm.

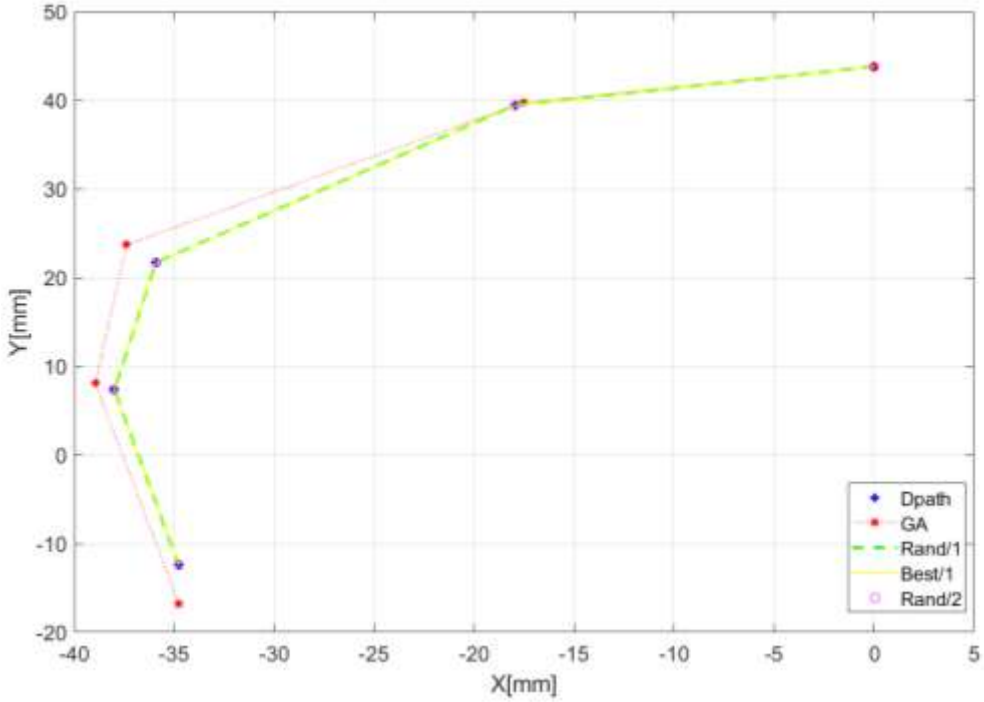

Fig. 7. Comparison between the best results obtained and desired points.

Based on plots of Figure 7 to indicate the desired path (Dpath) some blue markers are used. As can be seen, the trajectory obtained by the optimization process using GA (Espinosa-Garcia et al., 2021) is not closer to the Dpath. Referring to the best options obtained with different mutation variations are closer to the Dpath. The best result obtained by rand/1 is shown in color green, in color yellow the best obtained used best/1 is shown. Likewise, for the best result obtained by rand/2 strategy purple markers are used. On the other hand, it is important to mention that the parameters and sources to find these solutions were less than the presented in (Espinosa-Garcia et al., 2021), Table 6 shows a comparative.

Table 6. Comparison between parameters.

	Generations	Crossover value	Mutation value	Population
GA	1000	0.6	0.1	200
DE	300	0.2	0.1	100

Considering the parameters shown in Table 2, with the DE algorithm, the population parameter was reduced by 70% and the population by 50.

5 Conclusions

The differential evolutionary (DE) algorithm is a heuristic approach that compared with other heuristic methods like Genetic Algorithms (GA), particle swarm optimization (PSO) or Artificial Bee Colony algorithm (ABC) has advantages, DE found the

true global minimum regardless, fast convergence, and fewer control parameters. In this work these features are shown through two non-lineal optimization problems. The first example was a common test function (Rastrigin function), used to generate approximations. For this case, 15 tests were carried out, considering 5 for each mutation strategy. The results show feasible solutions, where the accuracy is determined for the mutation strategy implemented. Fortunately, in a test using the best/1 mutation strategy a fitness function with zero value was found, being this result, the best solution. Here it can be seen the potential of this algorithm because in other works do not find an exact solution, all of them are just approximations.

In addition, an optimization example of a mechanism for a robotic finger design is presented. This example is more complex because some restrictions must be satisfied to generate a feasible solution. In the original work the authors used a genetic algorithm as a method for solving the problem, but the obtained results have an error of 0.9908mm. But in this paper, using the results of the DE algorithm, the results obtained are better. It can be proved in Figure 7 because all the trajectories are closer to the desired path. The trajectory generated by the GA also is presented in red color, but the solution presents a major error. Moreover, the initial parameters like generation number in original work is 1000 and in this work the parameter is 300. The same occurs with the number of population while in GA the parameter is 200, with DE algorithm the value is half, reducing the computational resources. In the other hand, as main advantage of the usage of these kind algorithms is the variety of the approximations generated, because all of them represent a possible solution of a mechanism, which selection will depend on the designer needs and precision of the task to develop.

References

Baresel, A., Sthamer, H., & Schmidt, M. (2002). Fitness function design to improve evolutionary structural testing. En *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation* (pp. 1329-1336).

Brenner, S. C., Owens, L., & Sung, L. Y. (2008). A weakly over-penalized symmetric interior penalty method. *Electronic Transactions on Numerical Analysis, 30*, 107-127.

Espinosa-Garcia, F. J., Tapia-Herrera, R., Lugo-González, E., & Arias-Montiel, M. (2021). Development of a robotic hand based on a palm with a metamorphic mechanism for extending the thumb's functionality. *Journal of the Brazilian Society of Mechanical Sciences and Engineering, 43*(8), 404.

Fountas, N. A., & Vaxevanidis, N. M. (2020). Intelligent 3D tool path planning for optimized 3-axis sculptured surface CNC machining through digitized data evaluation and swarm-based evolutionary algorithms. *Measurement: Journal of the International Measurement Confederation, 158*, 107678.

Fountas, N. A., & Vaxevanidis, N. M. (2020). Intelligent 3D tool path planning for optimized 3-axis sculptured surface CNC machining through digitized data evaluation and swarm-based evolutionary algorithms. *Measurement: Journal of the International Measurement Confederation, 158*, 107678.

Kim, S. J., & Kim, J. H. (1993). Finite element analysis of laminated composites with contact constraint by extended interior penalty methods. *International Journal of Numerical Methods in Engineering, 36*(20), 3421-3439.

Lav, A. H., Goktepe, A. B., & Lav, M. A. (2009). Backcalculation of flexible pavements using soft computing. En K. Gopalakrishnan, H. Ceylan, & N. O. Attoh-Okine (Eds.), *Intelligent and Soft Computing in Infrastructure Systems Engineering* (Studies in Computational Intelligence, vol. 259). Berlin, Heidelberg: Springer.

Mallipeddi, R., Suganthan, P. N., Pan, Q. K., & Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing, 11*(2), 1679-1696.

Nozohour-leilabady, B., & Fazelabdolabadi, B. (2016). On the application of artificial bee colony (ABC) algorithm for optimization of well placements in fractured reservoirs; efficiency comparison with the particle swarm optimization (PSO) methodology. *Petroleum, 2*(1), 79-89.

Oguntola, M. B., & Lorentzen, R. J. (2021). Ensemble-based constrained optimization using an exterior penalty method. *Journal of Petroleum Science and Engineering, 207*, 109165.

Padmanabhan, S., Srinivasa, R. V., Asokan, P., Arunachalam, S., & Page, T. (2011). Design optimization of bevel gear pair. *International Journal of Design Engineering, 4*(4), 364-393.

Padmanabhan, S., Srinivasa, R. V., Asokan, P., Arunachalam, S., & Page, T. (2011). Design optimization of bevel gear pair. *International Journal of Design Engineering, 4*(4), 364-393.

Shaikh, J. H., Jain, N. K., & Pathak, S. (2016). Investigations on surface quality improvement of straight bevel gears by electrochemical honing process. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 230*, 1242–1253.

Shaikh, J. H., Jain, N. K., & Pathak, S. (2016). Investigations on surface quality improvement of straight bevel gears by electrochemical honing process. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 230*, 1242–1253.

Tham, J. Y., Ranganath, S., Ranganath, M., & Kassim, A. A. (1998). A novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology, 8*(4), 369-377.

Valdez, F., & Melin, P. (2007). Parallel evolutionary computing using a cluster for mathematical function optimization. En *NAFIPS 2007-2007 Annual Meeting of the North American Fuzzy Information Processing Society* (pp. 598-603). IEEE.

Vikhar, P. A. (2016). Evolutionary algorithms: A critical review and its prospects. En *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication* (pp. 261-265). IEEE.

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft Computing, 22*, 387-408.

Zolpakar, N. A., Lodhi, S. S., Pathak, S., & Sharma, M. A. (2020). Application of multi-objective genetic algorithm (MOGA) optimization in machining processes. En K. Gupta & M. Gupta (Eds.), *Optimization of Manufacturing Processes*. Springer Series in Advanced Manufacturing. Springer.

Zolpakar, N. A., Lodhi, S. S., Pathak, S., & Sharma, M. A. (2020). Application of multi-objective genetic algorithm (MOGA) optimization in machining processes. En K. Gupta & M. Gupta (Eds.), *Optimization of Manufacturing Processes*. Springer Series in Advanced Manufacturing. Springer.

Zolpakar, N. A., Yasak, M. F., & Pathak, S. (2021). A review: use of evolutionary algorithm for optimization of machining parameters. *International Journal of Advanced Manufacturing Technology, 115*, 31-47.

Zolpakar, N. A., Yasak, M. F., & Pathak, S. (2021). A review: use of evolutionary algorithm for optimization of machining parameters. *International Journal of Advanced Manufacturing Technology, 115*, 31-47.