Article **Journal-Mathematical and Quantitative Methods**
December 2023, Vol.7 No.13 25-32

Historical interpolation repair: a new method for handling limits in differential evolution

Reparación por interpolación histórica: un nuevo método para el manejo de límites en la evolución diferencial

JUAREZ-CASTILLO, Efrén†*, DEL ANGEL-GERARDO, Rodrigo, HERNÁNDEZ-HERNÁNDEZ, Avelina and MARTÍNEZ-HERNÁNDEZ, Ángel Francisco

Universidad Tecnológica de la Huasteca Hidalguense

ID 1st Author: Efrén, Juarez-Castillo / **ORC ID:** 0000-0002-2136-2516, **Researcher ID Thomson:** AAS-5698-2020, **CVU CONAHCYT ID:** 344990

ID 1st Co-author: Rodrigo, Del Angel-Gerardo / **ORC ID:** 0009-0001-4446-7185

ID 2nd Co-author: Avelina, Hernández-Hernández / **ORC ID:** 0009-0007-8050-0783

ID 3rd Co-author: Ángel Francisco, Martínez-Hernández / **ORC ID:** 0009-0006-9009-5593

DOI: 10.35429/JMQM.2023.13.7.25. Received July 25, 2023; Accepted December 30, 2023

Abstract

Differential Evolution (DE) is a population algorithm widely used in the solution of numerical optimization problems in continuous spaces. Sometimes, after mutation, DE generates solutions that exceed the search space, a requirement that justifies the implementation of Boundary Management Methods (BLM). This paper presents an innovative MML called "Historical Interpolation Repair", which relies on data from previous successful solutions to adjust those that deviate from the search space. Through a series of comparative experiments, it is shown that this approach outperforms other boundary handling methods. In fact, even under the most unfavorable conditions, the proposed method maintains high performance, suggesting that it is a solid choice for boundary management in DE.

Differential Evolution, Numerical Optimization, Boundary Handling, Boundary Constraints

Resumen

La Evolución Diferencial (ED) es un algoritmo poblacional ampliamente utilizado en la solución de problemas de optimización numérica en espacios continuos. En ocasiones, después de la mutación, ED genera soluciones que exceden el espacio de búsqueda, requerimiento que justifica la implementación de los Métodos para el Manejo de Límites (MML). Este artículo presenta un innovador MML llamado "Reparación por Interpolación Histórica", que se basa en los datos de soluciones exitosas anteriores para ajustar las que se desvían del espacio de búsqueda. Mediante una serie de experimentos comparativos, se demuestra que este enfoque supera a otros métodos de manejo de límites. De hecho, incluso en las condiciones más desfavorables, el método propuesto mantiene un alto rendimiento, lo que sugiere que es una opción sólida para la gestión de límites en ED.

Evolución Diferencial, Optimización numérica, Manejo de límites, restricciones de límites

Citation: JUAREZ-CASTILLO, Efrén, DEL ANGEL-GERARDO, Rodrigo, HERNÁNDEZ-HERNÁNDEZ, Avelina and MARTÍNEZ-HERNÁNDEZ, Ángel Francisco. Historical interpolation repair: a new method for handling limits in differential evolution. Journal-Mathematical and Quantitative Methods. 2023. 7-13:25-.

* Author Correspondence (E-mail: juarezefren@gmail.com)
† Researcher contributing as first author.

Article **Journal-Mathematical and Quantitative Methods**
December 2023, Vol.7 No.13 25-32

Introduction

Evolutionary computation is a subfield of artificial intelligence, which includes heuristic search algorithms that are inspired by biological evolution and its processes, such as natural selection and genetics. Such algorithms simulate processes such as mutation, recombination (or crossover) and survival of the fittest (or selection).

Generally, evolutionary computational algorithms start from a set of possible solutions to a problem and 'evolve' them over many generations (Eiben & Smith, 2015). Those solutions that are most effective in each generation are selected and modified to constitute the next generation of possible solutions. This process is repeated until a sufficiently effective solution is found or until other termination criteria are met.

Evolutionary Algorithms (EA) have gained recognition by demonstrating their high ability to solve complex numerical optimisation problems, whether combinatorial or continuous (Zhang, Peng, Zhang & Wang, 2023). This has been achieved through the implementation of techniques such as genetic algorithms, genetic programming, evolutionary strategies and differential evolution, among others.

Despite their effectiveness, certain EAs, such as Differential Evolution, encounter a specific challenge when dealing with problems with boundary constraints on the decision variables. These algorithms were originally designed to operate in search spaces without boundary constraints.

Boundary management, i.e., the way in which the algorithm is kept within the valid ranges defined for each design variable in a continuous space (the search space), is an aspect that is still being explored. This mechanism is essential in algorithms such as Differential Evolution (DE) and Evolutionary Strategies (ES), since their variation operators (such as crossover and mutation) do not ensure that the generated solutions stay within the bounds of the search space.

To address this problem, in algorithms such as ED and EE, boundary management methods are used. These methods are responsible for fitting solutions that lie outside the search space. In recent years, several studies have been published that reveal how the choice of boundary handling method can have a major impact on the quality of the solutions generated by the DE algorithm (Nocedal & Wright, 2006).

It is understood that there is no universal method for boundary handling that is optimal for all problems. However, this is seen as an opportunity to develop new methods that can improve the performance of the DE algorithm. With this in mind, this paper proposes a new method called "Historical Interpolation Repair", which uses information from previous successful solutions to adjust solutions that lie outside the search space. This method has been designed to improve the handling of boundary constraints in the context of algorithms such as Differential Evolution (DE).

Problem statement

Numerical Optimization Problems (NOP) arise in a wide range of situations in science and engineering. Their objective is to determine the values of a specific set of decision variables, through the optimisation of an objective function, provided that both the limits set for the variables and the functional constraints are met (Storn & Price, 1997; Eberhart, Kennedy et al., 1995). This type of problem is defined as follows:

$$Minimizar: f(\vec{x}) \tag{1}$$

Subject to:

$$l_j \leq x_j \leq u_j \tag{2}$$

Where $\vec{x} = [x_1, x_2, \ldots, x_n]^T \in R^n$ represents a solution vector and f($\vec{x}$) is the objective function.

Equation 2 sets the bounds on the variables in the solution vector, which restrict the possible values for the decision variables. Each variable x_j in $\vec{x}$ is bounded by a lower and an upper bound, which together define the search space S.

ISSN-On line: 2531-2979

Numerical Optimisation Problems are often tackled by traditional mathematical methods, although these may be insufficient for large-scale or complex problems. In contrast, Evolutionary Algorithms such as Differential Evolution have proven to be powerful alternatives, especially in situations where detailed information about the objective function is limited or uncertain. This article focuses on the application and analysis of Differential Evolution to solve this kind of problems, detailing its implementation in the following section.

Differential evolution

Differential evolution (DE) is a population-based stochastic search algorithm for solving optimisation problems in continuous spaces (Storn & Price, 1997). The DE algorithm aims to evolve a population of n-dimensional NP-vectors representing candidate solutions in the generation g:

$$\vec{x}_i^{\,g} = \{x_{i,1}^g,\ x_{i,2}^g, \dots, x_{i,n}^g\}, i = 1, \dots, NP \qquad (3)$$

Where $x_{i,j}$ represents the j-th component of solution i. The initial population is generated uniformly at random within the predefined lower l_j and upper u_j limits for each variable x_(i,j). The DE consists mainly of applying three operations that are repeated generation after generation until a termination criterion is satisfied. These operations are described below.

Mutation operator. For each target vector or "parent vector" x ⃗_i^(g-1) in generation g-1, a mutant vector v ⃗_i^gen generation g is created. The mutant vector can be created by some mutation strategy such as DE/rand/1/bin, described in Equation 4.

$$\vec{v}_i^{\,g} = \vec{x}_{r1}^{\,g-1} + F \times \left(\vec{x}_{r2}^{\,g-1} - \vec{x}_{r3}^{\,g-1}\right) \qquad (4)$$

Where *r1≠r2≠r3≠i* are indices generated within the range [1,NP]. F>0 is a real value representing a mutation scaling factor. Other DE variants exist, such as DE/best/1/bin and DE/rand/1/exp (Mezura-Montes, Miranda-Varela, & Gómez-Ramón, 2010), however, the present study focuses only on the DE/rand/1/bin variant.

The mutation operator can generate vector values outside the limits of each variable. Therefore, it is necessary to apply some method for handling boundary constraints in order to keep the search within the space defined by the bounds of each design variable.

Crossover operator. The crossover operator generates a test vector (child) by recombining the target vector and its corresponding mutant vector, as shown in Equation 5.

$$u_{i,j}^g = \begin{cases} u_{i,j}^g & \text{if}\left(\text{rand}_j[0,1] \le \text{CR}\right) \text{ or } \text{j} = j_{rand,} \\ x_{i,j}^g & \text{de lo contrario} \end{cases} \qquad (5)$$

Where $j = 1, \dots, n,\ CR \in [0,1]$ is a user-defined value indicating how similar the test vector will be with respect to the mutant vector, $randj$ is a randomly generated real value between 0 and 1, and $jrrand$ is a randomly generated integer within the range [1,n], its purpose is to avoid duplicates between the target vector and the test vector.

Selection operator. Finally, the selection operator is summarised in Equation 6, where the best vector is selected based on the fitness values between the target vector and its corresponding test vector. The vector with the best fitness value will remain in the next generation.

$$\vec{x}_i^{\,g} = \begin{cases} \vec{u}_i^{\,g} & \text{if } f\left(\vec{u}_i^{\,g}\right) < f\left(\vec{x}_i^{\,g-1}\right), \\ \vec{x}_i^{\,g-1} & \text{de lo contrario} \end{cases} \qquad (6)$$

The Differential Evolution algorithm has consistently demonstrated its effectiveness in solving complex engineering problems and applications related to numerical optimisation in continuous spaces. However, as previously indicated, it experiences challenges in handling boundary constraints. It is common for its mutation operator to generate solutions beyond the defined search space, S, which implies the need to incorporate boundary constraint handling methods.

Boundary constraint handling methods

Boundary Constraint Handling Methods are used to ensure that all components of the solution vectors lie within the lower and upper bounds of the decision variables.

ISSN-On line: 2531-2979

These methods are fundamental in DE and other evolutionary algorithms to maintain population feasibility and convergence to optimal solutions. The choice of these methods can significantly influence the quality of the solutions produced. The following is a description of commonly used methods for boundary handling in DE.

Wrapping:

In this method, the search space is wrapped in each dimension (Purchla, Malanowski, Terlecki, & Arabas, 2004), i.e. the search space of each variable is simulated to have a periodic shape and can therefore be treated as a toroidal space (Zhang, Xie, & Bi, 2004). For this reason, values leaving the search space at the upper boundary are inserted at the lower boundary through Equation 7.

$$x_j^c \begin{cases} x_j & si\ l_j < x_j < u_j \\ u_j - (l_j - x_j)\%\ p & si\ x_j < l_j \\ l_j + (x_j - u_j)\%\ p & si\ x_j > u_j \end{cases} \quad (7)$$

Where % is the modulus operator and 〚 p=‖u〛 _j-l_j | represents the range of the variable.

Reflection:

In this method, variables that violate limits (upper or lower) are reflected from the violated limit, by the number of violations (Robinson & Rahmat-Samii, 2004; Ronkkonen, Kukkonen, & Price, 2005). This method is described in Equation 8.

$$x_j^c = \begin{cases} x_j & si\ l_j \le x_j \le u_j \\ 2 \times l_j - x_j & si\ x < l_j \\ 2 \times u_j - x_j & si\ x_j > u_j \end{cases} \quad (8)$$

Where x_j^c is the corrected value, x_j is the value that violates some boundary constraint, l_j and u_j are the lower and upper bounds of variable j, respectively. This process is repeated until the generated value is within the bounds.

Boundary method:

In this method, also known as Bound or Projection, the value of the variable leaving the search space is reset to the violated bound (Brest, Greiner, Boskovic, Mernik & Zumer, 2006; Zhang, Xie, & Bi, 2004). This is expressed in Equation 9.

$$x_j^c \begin{cases} x_j & si\ l_j \le x_j \le u_j \\ l_j & si\ x_j < l_j \\ u_j & si\ x_j > u_j \end{cases} \quad (9)$$

In where x_j^c represents the corrected value x_j is the value that violates the bounds of the variable, l_j and u_j represent the lower and upper bounds of the variable respectively.

Random method:

This method replaces the values of the variables that are outside their bounds by random values within the lower and upper bounds (Lampinen, 2002; Price, Storn, & Lampinen, 2005) through Equation 10.

x_j^c=l_j+rand(0,1)×(u_j-l_j) (10)

Where rand(0,1) returns a random value between 0 and 1 with uniform distribution.

Centroid method:

The Centroid method relocates the invalid vectors within the search space to the centroid of an area formed by k+1 vectors, one of them taken from the population and k randomly repaired vectors (Juárez-Castillo, Pérez-Castro, & Mezura-Montes, 2017) by Equation 11.

The population vector biases the position of the corrected vector to an area with a good fitness value, while the k random vectors guide the position of the corrected vector to new areas in the search space.

$$\vec{x}^c = \begin{cases} \vec{x} & si \forall x_j : l_j \le x_j \le u_j \\ \frac{\vec{X}_{best} + \sum_{i=1}^{k} \vec{W}_{ri}}{k+1} & \text{en otro caso} \end{cases} \quad (11)$$

Where $\vec{x} = [x_1, x_2, \dots, x_n]$ is the vector that violates the bounds, n represents the number of dimensions of the problem to be treated, x ⃗ is the corrected vector, x ⃗_best is the best solution of the current population. On the other hand, W ⃗_r1,...,W ⃗_rk are k random vectors (Juárez-Castillo, Pérez-Castro, & Mezura-Montes, 2015). The random vectors initially take the same values as x ⃗, and subsequently replace their invalid values by randomly chosen values between their lower bound l_j and upper bound u_j, in the same way that a repaired vector is generated by the Random method described in Equation 10.

It is important to note that the Centroid method, originally proposed by Juárez-Castillo et al. (2019), was designed to address problems with functional constraints, where feasible and infeasible solutions exist. However, in this case, we are dealing with optimisation problems without functional constraints. Therefore, minor adjustments have been made to the Centroid method to make it suitable for the problems presented in this paper.

Res&Ran method:

In the Res&Ran method, when the Differential Evolution (DE) mutation operator, described in Equation 4, produces an invalid solution, the mutation operator is reapplied with the expectation of generating a valid solution by recreating the random values r1≠r2≠r3. If a valid solution is not obtained, this procedure is repeated up to 3×n times, where n denotes the dimensionality of the problem. In case no valid solution vector is obtained after these iterations, the Random method is applied (Juárez-Castillo, Pérez-Castro, & Mezura-Montes, 2017), as described in Equation 10.

Proposed method:

In this paper we propose a new method for handling boundary constraints called "Repair by Historical Interpolation", this strategy uses historical knowledge of the best solutions found during the optimisation process to repair invalid solutions. While this involves keeping a memory of the best solution at each generation, this memory is commonly generated in order to generate the convergence graphs, so it does not involve keeping additional records.

The repair is performed by a weighted interpolation between two of the best past solutions, s_1 and s_2, those closest to the invalid vector, as shown in Equation 12.

$$\vec{x}^c = \begin{cases} \vec{x} & si \ \forall \ x_j : l_j \leq x_j \leq u_j \\ \alpha * s_1 + (1-\alpha) * s_2 : \ s_1, s_2 \ \epsilon \ \mathcal{BS} \\ en \ cualquier \ otro \ caso \end{cases} \quad (12)$$

Where BS represents all the best solutions in each generation:

$$\mathcal{BS} = \{bs^1, bs^2, \dots, bs^g\} \quad (13)$$

Such that 〖bs〗 ^g represents the best solution in generation g. s_1 and s_2 are elements of BS, such that s_1 is the closest solution to x ⃗ and s_2 the second closest solution to x ⃗. To obtain the distance from 〖bs〗 ^g to x ⃗, the Euclidean distance formula is used. α is an interpolation parameter that takes values between 0 and 1 and determines the relative weighting of s_1 and s_1 at the interpolated point x ⃗.

Materials and methods

An experiment was conducted to assess the performance of the proposed Historical Interpolation Repair method by contrasting it against six other methods in solving a set of optimisation problems. This analysis took into account the following guidelines:
35 independent runs of the Differential Evolution (DE) algorithm were performed to address a benchmark consisting of 10 unconstrained continuous space numerical optimisation problems. The problems analysed, all of them minimisation problems, were: Ackley, Beale, Griewank, Michalewicz, Rastrigin, Rosenbrock, Schwefel, Schwefel_222, Sphere and Styblinski-Tang, all in 10 dimensions.

In each run of the ED algorithm, 100 generations were performed with the DE/rand/1/bin strategy, with F=0.7, CR=0.8 and population size NP=50, following the recommendations of the relevant literature (Zhang & Sanderson, 2009; Ronkkonen, Kukkonen, & Price, 2005).

ISSN-On line: 2531-2979

The same scheme was applied for each of the seven boundary management methods analysed: Wrapping, Reflection, Centroid, Res&Ran, Random, Bound and the Historical Interpolation Repair method, which will be referred to hereafter as Historic.

Results

Table 1 shows the average fitness value achieved over the 35 independent runs, highlighting in bold the best result for each of the 10 optimisation problems.

Additionally, the penultimate column shows the resultant value (H) of the Kruskal Wallis statistical test, while the p-value is presented in the last column.

As can be seen in Table 1, the Historic method obtained the best performance in all the problems treated, and as can be seen from the data generated by the statistical tests, the difference between the means is statistically significant.

Función	Historic	Wrapping	Reflection	Centroid	Res&Ran	Random	Bound	Kruskal Wallis (H)	Valor (p)
Ackley	**5.142**	16.576	16.566	13.130	15.501	15.197	17.401	193.584	**4.40E-39**
Beale	**0.018**	0.067	0.063	0.051	0.064	0.054	0.077	56.614	**2.19E-10**
Griewank	**3.665**	46.001	41.692	18.516	34.256	33.376	63.149	200.854	**1.25E-40**
Michalewicz	**-168.831**	-117.633	-118.981	-130.908	-121.558	-123.796	-110.725	144.214	**1.29E-28**
Rastrigin	**35.346**	75.256	75.280	55.344	67.785	66.679	82.513	173.391	**8.58E-35**
Rosenbrock	**2049.931**	18759.780	15727.982	6364.576	12673.040	11746.685	30470.934	177.004	**1.47E-35**
Schwefel	**710.249**	2024.166	2042.246	2443.639	2195.647	2198.008	2201.260	150.529	**5.98E-30**
Schwefel_222	**0.722**	3.947	3.918	2.340	3.240	3.372	4.529	200.344	**1.60E-40**
Sphere	**0.676**	13.716	12.241	4.999	9.894	9.159	16.695	206.042	**9.81E-42**
Styblinski-Tang	**-369.369**	-282.325	-283.791	-286.678	-291.923	-290.518	-255.152	137.659	**3.13E-27**

Table 1 Comparison of the performance of 7 boundary management methods for DE on 10 optimisation problems, with summary results of Kruskal Wallis statistical tests.

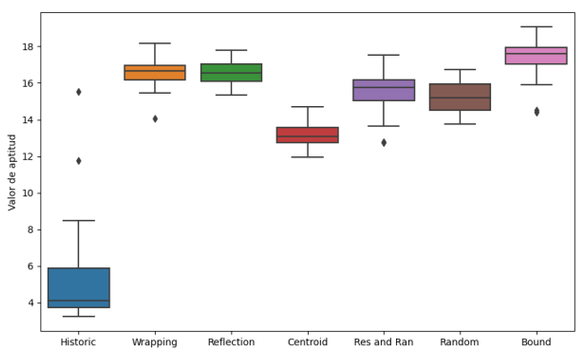

Figure 1 Comparison of methods for handling boundary constraints in DE: Distribution of fitness values obtained in 35 independent runs for the Ackley problem.

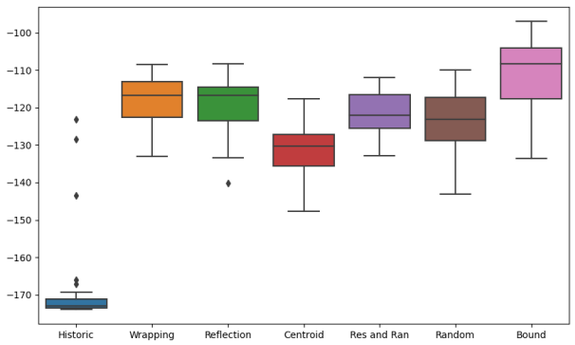

Figure 2 Comparison of methods for handling boundary constraints in DE: Distribution of fitness values obtained in 35 independent runs for the Michalewicz problem

After Historic, the second best performing method was Centroid, since it obtained the second best performance in 8 of the 10 problems treated, only being surpassed by Wrapping in the Schwefel problem and by the Res&Ran and Random methods in the Styblinski-Tang problem.

To complement the above, Figures 1 and 2 present two box plots illustrating the distribution of the fitness values achieved in the 35 independent runs for each of the 7 methods studied, in the Ackley and Michalewicz problems. For reasons of space, only 2 of the 10 problems evaluated are included, however, the rest of them show a similar trend.

In these graphs, a significant advantage of the Historic method for handling boundary constraints in the DE algorithm can be clearly observed when solving the Ackley and Michalewicz optimisation problems. In an optimisation scenario, a lower fitness value is preferable, which puts the Historic method in a prominent position. Not only does this method show a noticeably lower mean compared to the other methods, but also its highest values (represented by the upper "whiskers" in the box) are below the minimum values achieved by the other methods. This suggests an exceptionally consistent and efficient performance of the Historic method in terms of handling boundary constraints during optimisation, even in its worst-case performance scenarios. These observations highlight the effectiveness of Historic and set a strong precedent for its use in future DE applications.

ISSN-On line: 2531-2979

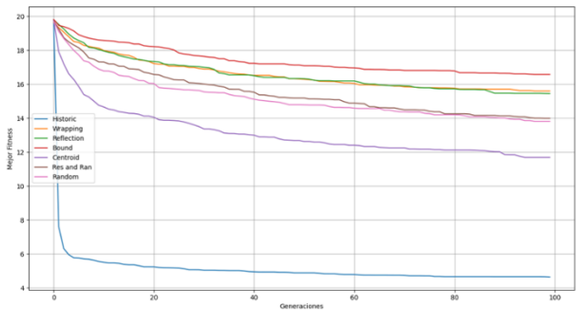

Figure 3 Comparison of Convergence of Methods for Handling Boundary Constraints in the Ackley Problem using Differential Evolution.

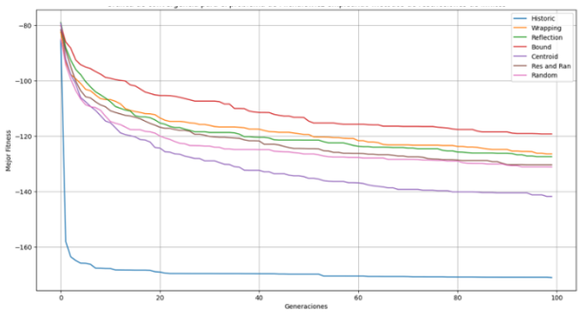

Figure 4 Comparison of Convergence of Methods for Handling Boundary Constraints in the Michalewicz Problem using Differential Evolution.

Finally, Figures 3 and 4 present two convergence plots for each of the seven methods compared in the Ackley and Michalewicz functions, which show the progress of the objective function over the generations for each method. Looking closely at these graphs, it is clear that the Historic method shows a much steeper downward trend than the other methods. Its rapid and steady decline indicates an effective convergence to optimal solutions. In addition, it is noted that the Centroid method was the second best, although its convergence rate is considerably slower compared to the Historic method.

In the last generation represented in the graphs, the results confirm the superiority of the Historic method over the others. While the other methods are in relative proximity in terms of objective function values, the Historic method stands out by being significantly below the other methods, thus demonstrating its effectiveness in obtaining more optimal solutions compared to the alternative approaches.

Conclusions

The handling of boundary constraints in Differential Evolution (DE) is a crucial challenge that affects the quality of solutions. This paper proposes a new method for handling boundary constraints, called "Historic Interpolation Repair" (Historic), which uses information from successful solutions to correct those that fall out of the search space, significantly improving the performance of DE.

The study's findings, based on tests covering ten numerical optimisation problems, indicate that the Historic method outperforms six methods tested, with its worst results still outperforming the best of the other methods, showing remarkable robustness and reliability.

The second best performing method was Centroid, which, like Historic, uses information from other solutions to guide the correction of invalid solutions.

The above seems to suggest that repair methods that employ some kind of information regarding the evolutionary process might show better performance, leaving open the possibility of exploring new, informed methods that employ information from solutions, either from the current population or from previous generations.

In summary, "Repair by Historical Interpolation" represents a breakthrough in the handling of boundary constraints in DE, improving its effectiveness and efficiency and providing valuable guidance for future proposals for repair methods.

References

Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE transactions on evolutionary computation, 10(6), 646-657.

Eberhart, R. C., Kennedy, J., et al. (1995). A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science (Vol. 1, pp. 39–43). New York, NY.

ISSN-On line: 2531-2979

Eiben, E., & Smith, J. E. (2015). Introduction to evolutionary computing (Natural Computing Series). Springer.

Juárez-Castillo, E., Pérez-Castro, N., & Mezura-Montes, E. (2015). A novel boundary constraint-handling technique for constrained numerical optimization problems. In 2015 IEEE Congress on Evolutionary Computation (CEC) (pp. 2034–2041). IEEE.

Juárez-Castillo, E., Pérez-Castro, N., & Mezura-Montes, E. (2017). An improved centroid-based boundary constraint-handling method in differential evolution for constrained optimization. International Journal of Pattern Recognition and Artificial Intelligence, 31(10), 1759023.

Lampinen, J. (2002). A constraint handling approach for the differential evolution algorithm. In Proceedings of the Congress on Evolutionary Computation (Vol. 2, pp. 1468–1473). IEEE Computer Society Washington, DC.

Mezura-Montes, E., Miranda-Varela, M. E., & Gómez-Ramón, R. d. C. (2010). Differential evolution in constrained numerical optimization: An empirical study. Information Sciences, 180(22), 4223–4262.

Nocedal, J., & Wright, S. (2006). Numerical optimization. Springer.

Price, K., Storn, R. M., & Lampinen, J. A. (2005). Differential evolution: A practical approach to global optimization (Natural Computing Series).

Purchla, M., Malanowski, M., Terlecki, P., & Arabas, J. (2004). Experimental comparison of repair methods for box constraints. In Proc. 7th National Conf. on Evolutionary Computation and Global Optimisation (pp. 135–142).

Robinson, J., & Rahmat-Samii, Y. (2004). Particle swarm optimization in electromagnetics. IEEE Transactions on Antennas and Propagation, 52(2), 397–407.

Ronkkonen, J., Kukkonen, S., & Price, K. V. (2005). Real-parameter optimization with differential evolution. In Proc. IEEE CEC (Vol. 1, pp. 506–513).

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11(4), 341–359.

Zhang, J., & Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. IEEE Transactions on evolutionary computation, 13(5), 945-958

Zhang, W. J., Xie, X. F., & Bi, D. C. (2004). Handling boundary constraints for numerical optimization by particle swarm flying in periodic search space. In Evolutionary Computation, 2004. CEC2004. Congress on (Vol. 2, pp. 2307–2311). IEEE.

Zhang, X., Peng, H., Zhang, J. y Wang, Y. (2023). Una red convolucional temporal de atención sensible a los costos basada en una evolución diferencial adaptativa top-k para una clasificación desequilibrada de series temporales. Sistemas expertos con aplicaciones, 213 , 119073.
.

ISSN-On line: 2531-2979