University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)

ISSN: 2714-3627

A Journal of the Faculty of Computing, University of Ibadan, Ibadan, Nigeria

Volume 16 No. 1, January 2026

journals.ui.edu.ng/uijslictr
http://uijslictr.org.ng/
uijslictr@gmail.com

University of Ibadan
Journal of Science and Logics in ICT Research

Modified Poll Search Differential Evolution Algorithm for Global Optimisation

Sawyerr B. A. *[1] **bsawyerr@unilag.edu.ng**
Fasina E. P.[1] **efasina@unilag.edu.ng**
Ajose A. M.[2] **aishatajosee@gmail.com**
Ojiako C. P.[1] **cojiako@unilag.edu.ng**

[1]*Department of Artificial Intelligence and Robotics*
Faculty of Computing and Informatics
University of Lagos
[2]*Department of Computer Sciences*
Faculty of Computing and Informatics
University of Lagos

Abstract
This paper introduces MPS-DE, an innovative variant of Differential Evolution (DE) that integrates the Modified Poll Search (MPS) technique with DE. The incorporation of MPS seeks to enhance the efficiency and effectiveness of the DE algorithm. A comprehensive empirical comparison was conducted among MPS-DE, RCGA-P, SRCGA, and DE/Best/1/Bin on a suite of twenty global optimisation benchmark functions. The results show that MPS-DE demonstrates superior performance compared to DE/Best/1/Bin and SRCGA, although RCGA-P outperforms MPS-DE in certain benchmarks, indicating the impressive and potential search capabilities of MPS in evolutionary algorithms.

Keywords: Differential Evolution, Real-Coded Genetic Algorithm, Global Optimisation, Modified Poll Search.

1. Introduction
Optimisation plays a crucial role in numerous scientific and engineering disciplines, as it enables us to discover optimal solutions to complex problems [1], [2]. Covering a broad array of subjects, optimisation can be demanding, prompting researchers to develop various algorithms, such as evolutionary algorithms and swarm intelligence methods, in tackling these challenges [3].

In the last five decades, evolutionary algorithms (EAs) have gathered significant interest, primarily due to their effectiveness in optimising very complex numerical functions [4]. Typically, EAs start with an initial population of randomly generated potential solutions. This population serves as the basis for generating subsequent generations of solutions through iterative evolutionary processes, which continue until a satisfactory solution is identified [5]. Leading examples of evolutionary algorithms include genetic algorithms, differential evolution, evolution strategies and genetic programming.

Differential Evolution (DE) is a simple and powerful type of evolutionary algorithm developed by Storn and Price in 1995 [6]. DE is a deterministic algorithm for solving numerical continuous problems. It makes use of mutation, crossover and selection operations to get an optimal solution [6], [7], [8], [9]. DE is less computationally demanding than other EAs and reaches solutions more quickly [1], [6].

DE has been tested with great success on many real-world problems, such as power systems, scheduling, etc [5], [9]. One of the main reasons DE is widely used is its adaptability and versatility and as a result, DE has been modified numerously by scientists to improve its performance [10], [11], [12], [13], [14].

The performance of the DE relies greatly on the choice of the operations and control parameters. The main DE operation that gets modified is the mutation operation. The mutation procedure is the most important operation in the DE process because it creates the individuals in the new populations. DE uses three control parameters:

Sawyerr B. A., Fasina E. P., Ajose A. M. and Ojiako C. P. (2026). Modified Poll Search Differential Evolution Algorithm for Global Optimisation, *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR),* Vol. 16 No. 1, pp. 192 – 201

population size (NP), crossover rate (CR), and scaling factor (F) [9], [15].

Although differential evolution is a popular global optimisation algorithm, it does have certain limitations, like other evolutionary algorithms [11]. This study introduces an enhanced version of DE that incorporates a local search method. The goal of this new variant is to improve both the convergence speed and overall efficiency of the original DE algorithm [11], [12], [13].

The remainder of the paper is divided into the following sections. Section 2 provides a literature review of DE, SRCGA and SRCGA-P. Section 3 presents the new MPS-DE, providing complete descriptions of the proposed modifications. Section 4 describes the experimental settings employed in the study. Section 5 presents the results and discussion, and finally, Section 6 concludes the paper.

2. Literature Review

2.1 Genetic Algorithm

The genetic algorithm developed by John Holland in 1975 is a meta-heuristic algorithm inspired by evolution. This method integrates an advanced global exploration approach with a survival-of-the-fittest mechanism that operates among individual vectors within the population [3], [16], [17].

The genetic algorithm begins with a population of randomly selected individuals. This population undergoes mutation and recombination processes, generating new individuals, and the genetic cycle is iteratively repeated through multiple generations. Through this iterative improvement, solutions are continually refined until a termination criterion is met. Various operators for selection, recombination, mutation, and elitism have been implemented to establish various variants of the genetic algorithm [5], [18], [19].

Algorithm 1: Standard Genetic Algorithm [16]

1. Generate **N** random and uniformly distributed solutions.
2. Repeat the following steps until the terminal requirement is met.
   a. Select **m** (**m** <= **N**) solutions in the population and save them into **P**(t)
   b. To produce offspring solutions, choose pairs of parents from **P**(t) and use an arithmetic crossover with probability.
   c. Create a new solution **m** by subjecting each bit of the offspring to mutation using a very low probability.
   d. Update the parent population by replacing **m** parents with the new individuals.
   e. Apply the elitism function to the new population

2.2 Projection-Based Real Coded Genetic Algorithm

The RCGA-P algorithm is based on the canonical real-coded genetic algorithm (RCGA). It includes the projection-based exploration operator in the canonical RCGA. By applying vector projection within RCGA, the algorithm improves the exploration search for solutions in the entire solution space by centrifugally searching the neighbourhood of the mutated offspring. For every element found in the mutated offspring, another element is randomly chosen, and the two are projected onto each other according to their fitness values [16], [20].

Algorithm 2: Projection-Based Genetic Algorithm [16], [21]

1. Generate **N** random and uniformly distributed solutions.
2. Repeat the following until the terminal requirement is met.
   a. Select **m** (**m** <= **N**) solutions in the population and save them into **P**(t)
   b. To produce offspring solutions, choose pairs of parents from **P**(t) and use an arithmetic crossover with probability.
   c. Create a new solution **m** by subjecting each bit of the new offspring to a low probability of selection.
   d. for each element in the mutated offspring, select a random individual in the population, if the fitness value of the mutated offspring is less than that of the random individual, then the mutated offspring is projected on the random individual, else the random individual is projected on the mutated string, and the result is added as a child for the next generation.
   e. Update the parent population by replacing **m** parents with the new individuals.

2.3 Differential Evolution

The differential evolution algorithm is a stochastic procedure for minimising a group of continuous functions that are dependent on a

population by using selection, mutation, and crossover operators to direct the population towards optimal solutions.

The Differential evolution was developed by Storn and Price in 1995 [6], and it is now a widely used optimisation algorithm for continuous optimisation problems. They suggested using a differential operator instead of the traditional crossover and mutation operators. By comparing DE to other evolutionary methods, it has a lower computational complexity, quicker convergence, and utilises computer memory better [1]. It has been used to solve many types of real problems, including communication, signal processing, electrical power, image processing, bioinformatics, control systems, pattern recognition, electromagnetism, chemical engineering, mechanical engineering, and many others, because of its growing popularity as an easy and very powerful optimiser [22]. A key benefit of differential evolution is that it is straightforward and only uses important control parameters, which are the population size and scale factor. The trial vector generation strategy and the control parameter selection significantly impact differential evolution's output in a specific optimisation problem [15], [22], [23].

2.3.1 Initialization

Differential evolution starts by populating the initial population with vectors that cover the entire parameter space. Generally, the initial population's randomised selection obeys a uniform probability distribution. Each member of the population is represented as an N-dimensional vector variable.

$$x_{i,t} = (x_{i,t}^1, x_{i,t}^2, \ldots, x_{i,t}^N), \text{ i = 1, 2, ..., NP} \quad (1)$$

where N is the problem dimension, t represents the current generation, and NP represents the population size [4], [6].

2.3.2 Mutation

Following the initialisation stage, the mutation operation, which is the most important step in differential evolution, is used to develop a new offspring in the population. DE's mutations are labelled $DE/x/y/z$: x represents the modified solution; it can be "random" or "best," and y indicates the number of difference vectors used to alter x. Each difference vector represents the distinction between two randomly selected but distinct members of the population. The recombination operator used (binomial or exponential) is represented by z [4], [6]. The scaled difference of two randomly selected vectors is added to a third parameter vector to produce a trial mutated vector. The scale factor F is a positive number that regulates the population's evolution rate. Although F has no maximum limit, efficient values rarely exceed 1.0 [24].

$$y_{i,t+1} = x_{rn1,t} + F * (x_{rn2,t^t} - x_{rn3,t}) \quad (2)$$

where rn1, rn2, and rn3 are all unique numbers between 1 and N.

Mutation is a significant phase in differential evolution, which creates new solutions in the population. Several studies have worked in this direction and suggested changes to mutation strategies [5]. The following are some of the widely used mutation strategies: DE/rand/1/exp, DE/rand/1/bin, DE/best/1/exp, DE/best/1/bin, DE/rand-to-best/1/exp, DE/rand-to-best/1/bin, DE/rand/2/exp, DE/rand/2/bin, DE/best/2/exp, DE/best/2/bin [6], [7].

2.3.3 Crossover

Discrete recombination creates trial vectors by comparing values from two different mutant vectors. The crossover probability, Cr, is a parameter value that determines how many parameter values are copied from the mutated vectors.

Uniform crossover uses a random number and the crossover probability to determine which parent provides each parameter. If the random value falls below the crossover probability, the trial parameter is taken from the mutated vector; if not, it is taken from the target vector.

$$y_{i,j} = \{ y_{i,j} \ if \ rand(0,1) < crossover \ probability \ x_{i,j} \ otherwise \quad (3)$$

The crossover operator uses a mutated vector to create a new trial vector. Storn and Price [6] originally recommended the exponential crossover, but the binomial version gained acceptance among academics [5].

2.3.4 Selection

Trial vectors that have fitness values equal to or less than the target vectors replace the target vectors in the population for at least one more generation, while trial vectors that have fitness

values greater than the target vectors do not replace the target vectors [4].

$$x_{i,t+1} = \{y_{i,j} \ \ if\ f(y_{i,j}) < f(x_{i,j}) \quad x_{i,j}\ otherwise \tag{4}$$

Although only a few papers propose changes to the selection scheme, researchers have found that suitable modifications can further improve the algorithm's performance [5].

2.3.5 Termination

Differential evolution combines recombination and selection more closely than other Evolutionary Algorithms by contrasting each trial vector with the goal vector it inherits parameters. If the new population is in place, the mutation, recombination, and selection process is replicated until the best solution is found or a predetermined termination condition is met, such as the number of generations reaching a predetermined limit.

Algorithm 3: Differential Evolution Algorithm

1. Generate N random and uniformly distributed solutions.
2. Repeat while the maximum generation has not been exceeded or a solution hasn't been found for each target vector xi,t in the population:
   i. At random, choose 2 candidate vectors from the population: xrn1,t, xrn2,t. They must be unique from one another and from the target vector xi,t and the current optimal solution.
   ii. Using the mutation operation, create the mutated vector yi,t.
   iii. The elements of the target vector xi or the mutated vector yi,t are used to create the trial vector ui,t.
   iv. If any element ui,j in the trial vector ui,t is out of bounds, clip ui,j, so it stays within the specified boundary
   v. If the trial vector has a lower fitness value than the target vector, replace the target vector in the population with the trial vector.

3. Differential Evolution with Modified Poll Search

This algorithm was created to enhance differential evolution by helping it locate the optimal solution more effectively. This is a differential evolution algorithm that incorporates a modified poll search MPS [16] on the trial vector $u_{i,t}$ produced by the crossover operation. An improved solution point in the neighbourhood of the trial vector $u_{i,t}$ can be found by using the modified poll step. MPS accomplishes this by generating a poll point $v_{i,t}$ around the present solution $u_{i,t}$ using the formula below.

$$v_{i,t} = u_{i,t} + r * U \tag{5}$$

where $u_{i,t}$, the trial vector is gotten from the crossover operation, r = ηΔ_t, is a step size and U = $(U_1, U_2, \ldots, U_n)^T$ is a directional cosine with random components.

$$U_k = \frac{R_k}{(R_1^2 + R_2^2 + \ldots + R_N^2),}, \ k = 1,2,\ldots,N \tag{6}$$

$R_k \sim$ Unif ([−1, 1]) and η is a step factor. Sometimes, the trial point components produced during the search will fall outside of the specified boundaries. In these instances, the poll vector $v_{i,t}$ components are recreated using the formula below.

$$v_{i,t}^j = \{u_{i,t}^j + \lambda * (ub_{i,t}^j - u_{i,t}^j), \ \ if\ v_{i,t}^j > ub_{i,t}^j \quad u_{i,t}^j + \lambda * (u_{i,t}^j - lb_{i,t}^j), \ \ if\ v_{i,t}^j < lb_{i,t}^j \tag{7}$$

The poll search has a parameter Δ_t which is initialised at the beginning of the algorithm. At t = 0, Δ_t is initialised by

$$\Delta_0 = \tau \times \max\ \{ub^j - lb^j, j = 1, 2, \ldots, n\} \tag{8}$$

The value of Δ_t varies with each generation. The mean of q unique points in the parent solution is calculated, followed by the distances between the mean of the points and each point. After analysing all of the distances, the K solutions that are closest to the mean are picked and used to calculate Δ_{t+1}.

$$\Delta_{t+1} = \frac{1}{k}\sum_{i=1}^{k} \quad y^i \tag{9}$$

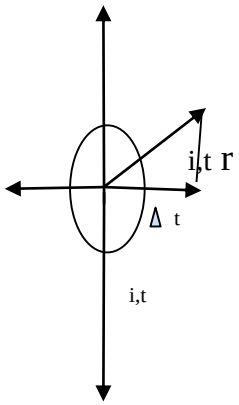


Figure 1: The generation of a trial point, $y_{i,t}$, by the POLL step.

Algorithm 4: Differential Evolution with Modified Poll Search

1. Generate N random and uniformly distributed solutions.
2. Repeat while the maximum generation has not been exceeded or a solution hasn't been found, Do the following for each target vector $x_{i,t}$ in the population:
   i. At random, choose 2 candidate vectors from the population: xrn1,t, xrn2,t. They must be unique from one another and from the target vector xi,t and the current optimal solution.
   ii. Using the mutation operation, create the mutated vector yi,t.
   iii. Using the crossover operation, create the trial vector ui,t.
   iv. Using the modified poll step to generate vi,t from ui,t
   v. If any element vi,j in the poll vector vi,t is out of bounds, clip vi,j, so it stays within the boundary
   vi. If the poll vector has a lower fitness value than the target vector, replace the target vector in the population with the poll vector.

4. Experimental Setting

A suite of 20 benchmark optimisation functions was used to test the performance of the hybrid differential evolution (MPS-DE). Comparative analyses were conducted with the standard real-coded genetic algorithm (SRCGA), the projection-based genetic algorithm (SRCGA-P), and the differential evolution algorithm (DE/Best/1/Bin). Due to the stochastic nature inherent in all evolutionary algorithms, each method was executed 15 times across the benchmark functions.

4.1 Parameter Selection

Table 1: Parameter Selection for the Genetic Algorithms

Parameter	**Value**
Population Size	50
Maximum number of generations	10000
Crossover probability	0.7
Mutation probability	0.001
Selection strategy	3 tournament selections

Table 2: Parameter Selection for the Differential Evolution Algorithms

Parameter	**Value**
Population Size	50
Maximum number of generations	10000
Mutation strategy	DE/best/1/bin
Scale factor	0.8
Crossover probability	0.9
r	0.2
Q	10
K	5

5. Experimental Results

The results of experiments using 20 benchmark functions on the SRCGA, RCGA-PS, DE/Best/2, and MPS-DE are shown in Table 3. Two key characteristics were used to evaluate the performance of the algorithms in this study. These characteristics are the algorithm's *convergence* and *robustness*.

An algorithm is said to be convergent if it converges to the desired result. A single execution of the algorithm is halted in this experiment once the algorithm converges to an acceptable solution. Execution of the algorithm has fulfilled its convergence property when the absolute difference between the best fitness value and the optimal solution of the benchmark function is less than or equal to a constant value called epsilon (ε). In this study, the value of epsilon is 10^4. Execution also stops when the maximum number of generations has been reached, and the algorithm does not converge [16].

Table 3: Comparison of the best fitness values f_{min} and success rate SR of SRCGA, SRCGA-P, DE and the proposed DE

			Best fitness score (F_{min})				**Success Rate (SR)**			
S/N	**Function name**	**f(x)**	**SRCGA**	**RCGA-P**	**DE**	**MPS-DE**	**SRCGA**	**RCGA-P**	**DE**	**MPS-DE**
1	Ackley	0.0	7.4868e-07	7.4777e-09	1.5944e-07	**2.8395e-09**	15	15	15	15
2	Rastrigin	0.0	1.452e-08	**1.3856e-13**	4,5086e-09	1.0544e-10	15	15	15	15
3	Rosenbrock	0.0	4.7067e-06	6.4678e-07	1.6913e-12	**2.2738e-13**	15	15	15	15
4	Schwefel	-837.9658	-837.96578	-837.9658	-837.96577	-837.96568	15	15	15	15
5	Sphere	0	2.2344e-14	**3.6397e-18**	5.0874e-16	1.0607e-17	15	15	15	15
6	AluffiPentini	−0.3523	-0.35223	-0.3523	-0.3523	-0.3523	15	15	15	15
7	Becker and Lago	0.0	1.2246e-13	**1.0797e-18**	1.1568e-16	3.64981e-17	15	15	15	15
8	Bohachevsky 1	0.0	3.1608e-12	0	6.6613e-16	1.1102e-16	15	15	15	15
9	Bohachevsky 2	0.0	6.5453e-13	**5.5511e-17**	4.3520e-14	**5.5511e-17**	13	15	15	15
10	Branin	0.3978	0.3978	0.3979	0.3978	0.3978	15	15	15	15
11	CamelBack 3	0.0	5.3290e-15	**0.0**	4.4409e-16	**0.0**	15	15	15	15
12	CamelBack 6	−1.0316	-1.0316	-1.0316	-1.0316	-1.0316	15	15	15	15
13	Cosine Mixture	−0.2000	-0.1999	**-0.2**	-0.1999	**-0.2**	15	15	15	15
14	Dekkers and Aarts	−24776.5183	-24776.5183	-24776.5183	-24776.5183	-24776.5183	15	15	15	15
15	Easom	-1.0	-0.9999	**-1.0**	-0.9999	**-1.0**	15	15	15	15

16	Goldstein and Price	3.0	3.0000	3.0000	3.0000	3.0000	15	15	15	15
17	Griewank	0.0	4.4828e-15	**1.4116e-21**	7.1531e-17	1.0028e-19	15	15	15	15
18	Shekel 5	−10.1532	-10.1532	-10.1532	-10.1532	-10.1532	10	12	10	15
19	Shekel 7	−10.4029	-10.4029	-10.4029	-10.4029	-10.4029	13	15	15	15
20	Shekel 10	−10.5364	−10.5364	−10.5364	−10.5364	−10.5364	11	15	14	15

The robustness of the algorithms is another key attribute used to evaluate their performance. A robust algorithm can solve a variety of problems and isn't limited to a single type of problem. The success rate of all the algorithm's runs is used to measure its robustness in this study. A single execution is said to be successful if it yields an optimal solution, that is, if the result of the execution is the same as the global optimal solution or if the absolute difference between the best fitness value and the benchmark function's optimal solution is less than or equal to epsilon(ε), in this case, 0.009 [16], [20]. The best fitness values (f_{min}) and the success rate (SR) are used in comparing the algorithms.

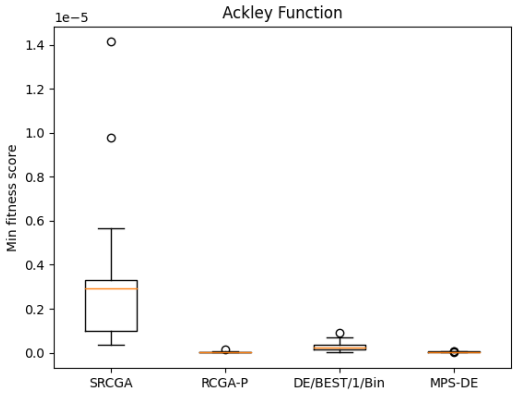
5.1 The Comparisons of modified DE with RCGA, RCGA-P and DE

This section briefly summarises the experimental outcomes shown in Table 3. The first two columns list the test problems and their names. The third column reports each benchmark's optimal value, followed by best fitness values and success rates for each algorithm. Table 3 and Figure 2's box plots serve as the basis for comparison among four algorithms.

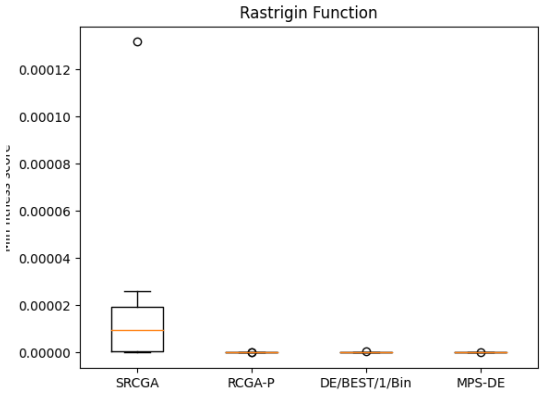
The purpose of the comparative analysis is to assess whether the proposed modification led to improved performance for DE. Examination of the results presented in Table 3 indicates that MPS-DE demonstrated superior outcomes relative to DE and SRCGA across most test cases; however, RCGA-P exhibited better results than MPS-DE on the Rastrigin, Sphere, Becker and Lago, Bohachevsky 2, and Griewank problems.

The success rate (SR) serves as a measure of an algorithm's robustness when addressing specific problem instances. According to Table 3, MPS-DE achieved the highest SR among all algorithms, with RCGA-P, DE, and SRCGA ranking subsequently.

An examination of the box plots presented in Figure 2 (i-vi) indicates that MPS-DE and RCGA-P consistently outperformed DE and SRCGA across all evaluated problems. Additionally, MPS-DE demonstrated superior performance to RCGA-P in addressing the Rosenbrock problem. Collectively, these results suggest that the MPS-DE algorithm is both efficient and robust, as evidenced by its consistent ability to locate the global minima for each of the 20 benchmark functions.



(i)



(ii)

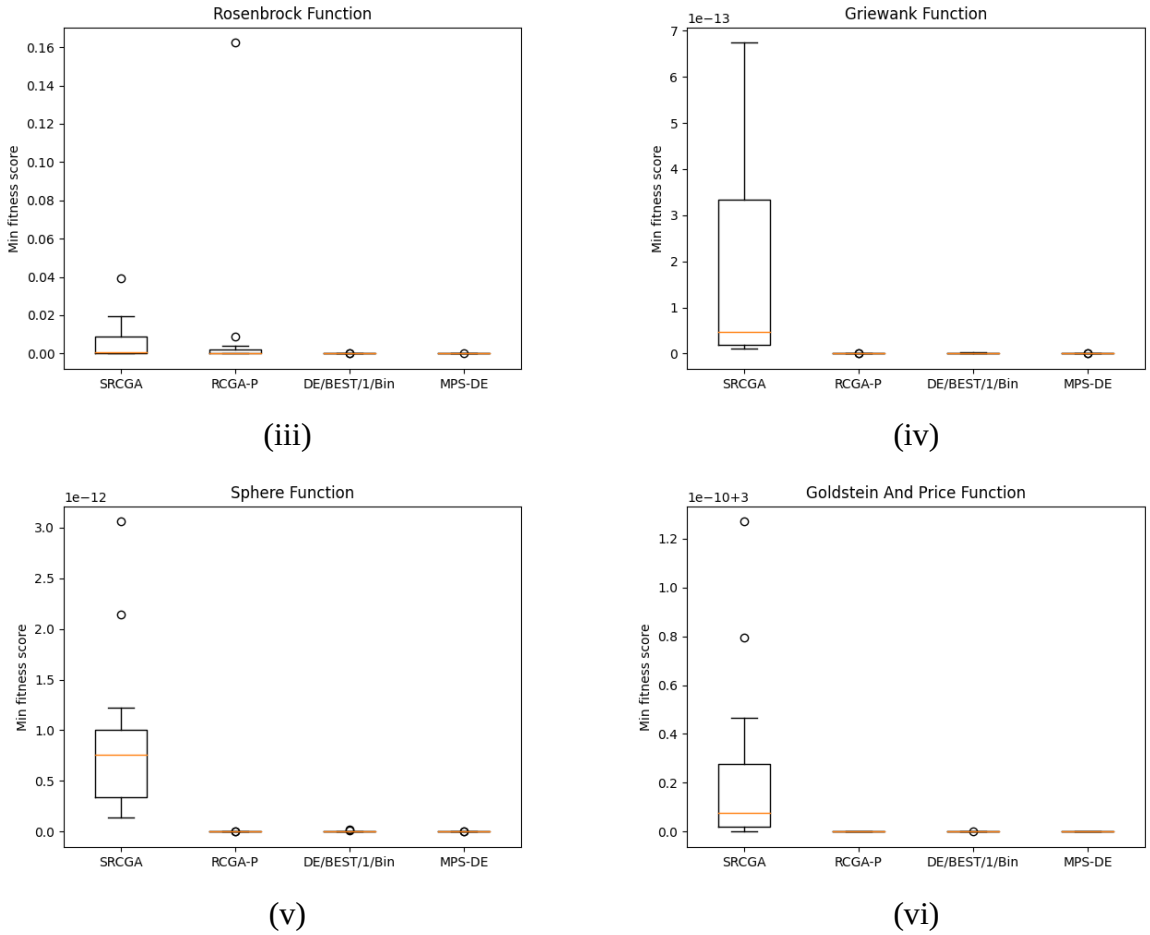


Figure 2: A group of box plots (i)–(vi) showing the differences between the minimum fitness value of the SRCGA, RCGA-PS, DE/Best/1 and MPS-DE on Ackley, Rastrigin, Rosenbrock, Griewank, Sphere, and Goldstein and Price, respectively.

6. Conclusion

In this study, the MPS-DE algorithm employed a modified poll search strategy for local exploration by examining the neighbourhood of the trial vector, and it identifies solutions that may outperform the original trial vector. The algorithm's performance was evaluated against SRCGA, RCGA-P developed by [16], [20], and the DE/BEST/1/BIN variant across 20 numerical benchmark functions. The results obtained indicate that MPS-DE consistently delivers higher-quality performance. Furthermore, the algorithm demonstrates greater robustness than its counterparts, achieving a success rate of 15 on most benchmark functions. There is a need to experiment with the parameters to get better performance of MPS-DE. MPS-DE has the potential to tackle several complex real-life optimisation problems.

References

[1] M. Georgioudakis and V. Plevris, "A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization," *Front. Built Environ.*, vol. 6, Jul. 2020, doi: 10.3389/fbuil.2020.00102.

[2] S. Akhmedova, V. Stanovov, D. Erokhin, and O. Semenkina, "Ensemble of the nature-inspired algorithms with success-history based position adaptation," in *IOP Conference Series: Materials Science and Engineering*, Institute of Physics Publishing, Jan. 2020. doi: 10.1088/1757-899X/734/1/012089.

[3] T. Alam, S. Qamar, A. Dixit, and M. Benaida, "Genetic algorithm: Reviews, implementations and applications," 2021, *Kassel University Press GmbH*. doi: 10.3991/IJEP.V10I6.14567.

[4] NOOR HUSSIEN ALI AWAD, "Numerical Optimization Using Differential Evolution," *PhD Thesis*, 2019.

[5] Bilal, M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, "Differential Evolution: A review of more than two decades of research," *Eng. Appl. Artif. Intell.*, vol. 90, p. 103479, Apr. 2020, doi: 10.1016/j.engappai.2020.103479.

[6] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution*. Berlin/Heidelberg: Springer-Verlag, 2005. doi: 10.1007/3-540-31306-0.

[7] E. Efr´, E. Mezura-Montes, J. Jes´, J. Veí Azquez-Reyes, and C. A. C. Coello, "A Comparative Study of Differential Evolution Variants for Global Optimization," 2006.

[8] D. Dawar, "ADAPTIVE DIFFERENTIAL EVOLUTION AND ITS APPLICATION TO MACHINE VISION," 2016.

[9] T. Eltaeib and A. Mahmood, "Differential evolution: A survey and analysis," *Applied Sciences (Switzerland)*, vol. 8, no. 10, Oct. 2018, doi: 10.3390/app8101945.

[10] S. K. Goudos, K. B. Baltzis, K. Antoniadis, Z. D. Zaharis, and C. S. Hilas, "A comparative study of common and self-adaptive Differential Evolution strategies on numerical benchmark problems," in *Procedia Computer Science*, 2011, pp. 83–88. doi: 10.1016/j.procs.2010.12.015.

[11] M. Chen, C. Feng, and and R. Cheng, "MetaDE: Evolving Differential Evolution by Differential Evolution," Mar. 2025, doi: 10.1109/TEVC.2025.3541587.

[12] Y. Shen, J. Wu, M. Ma, X. Du, and D. Niu, "Application of an Improved Differential Evolution Algorithm in Practical Engineering," *Concurr. Comput.*, vol. 37, no. 3, Feb. 2025, doi: 10.1002/cpe.8358.

[13] V. Stanovov and E. Semenkin, "Success Rate-based Adaptive Differential Evolution L-SRTDE for CEC 2024 Competition," in *2024 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, Jun. 2024, pp. 1–8. doi: 10.1109/CEC60901.2024.10611907.

[14] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu, "An improved differential evolution algorithm and its application in optimization problem," *Soft comput.*, vol. 25, no. 7, pp. 5277–5298, Apr. 2021, doi: 10.1007/s00500-020-05527-x.

[15] G. Jeyakumar and C. Shanmugavelayutham, "Convergence Analysis of Differential Evolution Variants on Unconstrained Global Optimization Functions," *International Journal of Artificial Intelligence & Applications*, vol. 2, no. 2, pp. 116–127, Apr. 2011, doi: 10.5121/ijaia.2011.2209.

[16] B. A. Sawyerr, M. M. Ali, and A. O. Adewumi, "A comparative study of some real-coded genetic algorithms for unconstrained global optimization," *Optim. Methods Softw.*, vol. 26, no. 6, 2011, doi: 10.1080/10556788.2010.491865.

[17] J. H. Holland, "Genetic Algorithms and Adaptation," in *Adaptive Control of Ill-Defined Systems*, Boston, MA: Springer US, 1984, pp. 317–333. doi: 10.1007/978-1-4684-8941-5_21.

[18] J. C. Felix-Saul, M. García-Valdez, J. J. Merelo Guervós, and O. Castillo, "Extending Genetic Algorithms with Biological Life-Cycle Dynamics," *Biomimetics*, vol. 9, no. 8, p. 476, Aug. 2024, doi: 10.3390/biomimetics9080476.

[19] B. A. Sawyerr, A. O. Adewumi, and M. M. Ali, "Real-coded genetic algorithm with uniform random local search," *Appl. Math. Comput.*, vol. 228, 2014, doi: 10.1016/j.amc.2013.11.097.

[20] B. A. Sawyerr, "Hybrid Real Coded Genetic Algorithms with Pattern Search and Projection," 2010.

[21] B. A. Sawyerr, A. O. Adewumi, and M. M. Ali, "Benchmarking projection-based real coded genetic algorithm on BBOB-2013 noiseless function testbed," in *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference Companion*, 2013. doi: 10.1145/2464576.2482698.

[22] D. Zaharie, "Influence of crossover on the behavior of Differential Evolution Algorithms," *Appl. Soft Comput.*, vol. 9, no. 3, pp. 1126–1138, Jun. 2009, doi: 10.1016/j.asoc.2009.02.012.

[23] P. Singal, A. C. Kumari, and P. Sharma, "Estimation of Software Development Effort: A Differential Evolution Approach," in *Procedia Computer Science*, Elsevier B.V., 2020, pp. 2643–2652. doi: 10.1016/j.procs.2020.03.343.

[24] T. Sum-Im, "A Novel Differential Evolution Algorithmic Approach to Transmission Expansion Planning," 2009.