A Modified Emperor Penguin Algorithm for Solving Stagnation in Multi-Model Functions

Ahmed Serag[1], Hegazy Zaher[2], Naglaa Ragaa[3], Heba Sayed[4]
[1]Operations Research, faculty of graduate studies for statistical research, Cairo University, Giza, Egypt
[2]Mathematical Statistics, Faculty of Graduate Studies for Statistical Research, Cairo University
E-mail: ahmede.serag1978@gmail.com, hgsabry@cu.edu.eg, naglaa777subkiii@yahoo.com, hmhmdss@yahoo.com

Keywords: optimization, metaheuristics, emperor penguin algorithm

Received: October 13, 2023

Abstract: Metaheuristic algorithms have gained attention in recent years for their ability to solve complex problems that cannot be solved using classical mathematical techniques. This paper proposes an improvement to the Emperor Penguin Optimizer algorithm, a population-based metaheuristic. The original algorithm often gets stuck in local optima for multi-modal functions. To address this issue, this paper presents a modification in the relocating procedures that allows the algorithm to utilize information gained from the previous positions of each penguin. To demonstrate the effectiveness of the modified algorithm, 20 test optimization functions from well-known benchmarks were selected. The implemented comparative analysis assesses the proposed algorithm against both the traditional Emperor Penguin Optimizer algorithm and one of the most recent algorithm modifications in current research. The findings indicate that the proposed algorithm demonstrates significant efficiency, particularly in addressing multi-modal functions, as evidenced by superior mean results and robustness.

Povzetek: Predstavljen je izboljšani algoritma "cesarskega pingvina", ki učinkovito rešuje kompleksne probleme zlasti v multimodalnih funkcijah.

1 Introduction

Problem-solving can be approached in various ways, such as trial and error, experimental design, or mathematical techniques [1]. Operations research is a field that employs mathematical formulation to solve complex engineering and management problems and gain insight into potential solutions [2]. Mathematical approaches used in this field can be classified into classical techniques like simplex method and dynamic programming, and heuristic and metaheuristic methods like the Emperor Penguin Optimizer algorithm (EPO) [3].

EPO is a population-based metaheuristic algorithm proposed by Dhiman & Kumar [4], inspired by the behavior of emperor penguins in utilizing crowds to survive the Antarctic winter. While the original algorithm is efficient in solving unimodal problems, it stagnates with complex problems like multi-local minima problems. Previous work on EPO in the literature includes the binary version by Dhiman et al. [5], multi-objective optimization by Kaur et al. [6], photovoltaic system optimization by Sameh et al. [7], support vector machine optimization for face recognition by Yang and Gao [8], RGB image threshold optimization by Jia et al. [9], and energy-efficient residential building design by Tang et al. [10]. Kaur et al. [11] adapted the emperor penguin optimizer algorithm to solve multi-objective optimization problems using the concept of dynamic archive. Xing [12] proposed an EPO algorithm for solving the multilevel threshold for color image segmentation. Lu et al. [13] improved the EPO algorithm by optimizing its output using the sequential quadratic programming. They used the improved algorithm to minimize the market clearing price probability function.

However, most of these studies focused on applying the EPO algorithm to solve real-world problems and did not address its stagnation issue in multi-modal problems. Table 1 summarizes the previously mentioned literature in this paper.

This paper proposes a modification to the EPO algorithm to increase its efficiency. The second section discusses the original algorithm's main steps, followed by the proposed modification in the third section. The fourth section presents comparative results between the original and modified EPO, and the fifth section concludes the paper.

2 The original emperor penguin optimizer algorithm

The Emperor Penguin Optimizer (EPO) is a population-based metaheuristic inspired by the crowd behavior of emperor penguins. The algorithm's steps include calculating the ambient temperature, distances toward the emperor penguins, and effective movers. To calculate the temperature around the crowd (TA), the temperature of each penguin (T) is considered, depending on the radius (R) that surrounds the crowd. The temperature profile around the crowd can be calculated using Equation (1), where the iteration number (Itr) and the maximum number of iterations ($MaxItr$) are taken into account.

$$TA = T - \frac{MaxItr}{Itr - MaxItr}, \forall Itr < MaxItr \quad (1)$$

Table 1: Literature summary

Author	Modification	Application
Dhiman and Kumar [4]	First Proposed	Test Optimization Problems
Jia et al. [9]	disruptive polynomial mutation Levy flight thermal exchange operator	Satellite image segmentation
Xing [12]	Gaussian mutation Levy flight	multilevel threshold for color image segmentation
Kaur et al. [6]		Muti-objective optimization
Yang and Gao [8]		Face recognition
Kaur et al. [11]		Multi-objective optimization
Lu et al. [13]	Sequential quadratic programming	Market clearing price
Dhiman et al. [5]	Binary emperor penguin optimizer	
Sameh et al. [7]		Photovoltaic control system
Tang et al. [10]		Energy consumption of the residential buildings

$$T = \begin{cases} 0, & if\ R > 1 \\ 1, & if\ R < 1 \end{cases}$$

The distance between the penguins and their emperor (D) is to be calculated using some sort of parameters related to avoiding collision between penguins (A), the position of the best penguin (P_{best}), the position of each penguin (P), the ambient temperature, TA, and the social force (S) that forces the penguins to move towards the direction of best solution. The parameter A is to be calculated for the position P_i using the movement parameter (M), which is set to 2, in equation (2).

$$A = M \times (TA + |P_{best} - P_i| \times rand) - TA \quad (2)$$

Equation (3) is used to calculate the social force. This equation is a decreasing function that has three variables, f, l, and Itr. Both of f and l are random numbers that each as its lower and upper bound. Figure 1 shows an example of having f belongs to the interval $[2, 5]$ and l belongs to the interval $[2, 10]$.

$$S = \left(\sqrt{f\ .e^{-Itr/l} - e^{-Itr}}\right)^2 \quad (3)$$

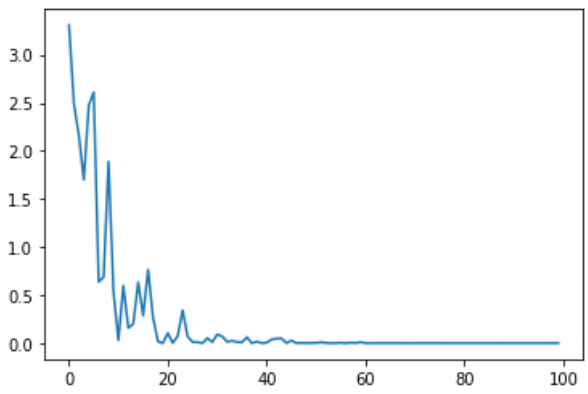

Figure 1 Social force parameter's graph

The parameter S is used to calculate the distance$D$, where it increases in the very first iterations to guarantee high locality and decrease it gradually until reaching very low locality in the higher iterations. So, the distance D is to be calculated using equation (4) as follows:

$$D = |S\ .\ P_i - rand\ P_{best}| \quad (4)$$

Now, the position of the penguin in the next iteration (P_{i+1}) can be calculated using equation

(5) as follows:

$$P_{i+1} = P_i - A\ .D \quad (5)$$

The best position is updated by changing the penguin's position in each iteration until the stopping criteria are met, and then the best solution is returned. The pseudocode for the algorithm can be summarized as follows:

Input the Populatin size (N), $MaxItr$, *and R parameters*
Generate the initial population
Evaluate each solution in population and store the best solution (P_{best})
$Itr = 1$
While $Itr \leq MaxItr$ *do*:
    $i = 1$
    While $i \leq N$ *do*:
        $TA = T - \frac{MaxItr}{Itr - MaxItr}$
        $A = M \times (TA + |P_{best} - P_i| \times rand) - TA$
        $S = \left(\sqrt{f\ .e^{-Itr/l} - e^{-Itr}}\right)^2$
        $D = |S\ .\ P_i - rand\ P_{best}|$
        $P_{i+1} = P_i - A\ .D$
        if $f(P_{i+1}) \leq f(P_{best})$ *then*:
            $P_{best} = P_{i+1}$
        $i = i + 1$
    $Itr = Itr + 1$
Return Gbest

3 The proposed modified EPO algorithm

In this section, a new modification to the EPO algorithm is proposed, which involves utilizing information gained from the current position of each penguin before relocating it. This modification constructs an information vector using a threshold selected from the interval $[0, 1]$. The purpose of this proposed mutation process is to achieve increased locality when the lower threshold is reached. This is done by creating a new solution with more components from the current solution. Conversely, when

the higher threshold is reached, the process aims to enhance diversity by incorporating more components from the relocated solution. The current position, P_i, and the position generated using equation
(5) are used to create the information vector, P_{IV}. The new procedure selects the components of P_{IV} using the threshold. The first step of the new procedure is to generate a uniform random number that will be compared to the selected threshold. If the generated uniform random number is greater than the threshold, then component j is selected from P_{i+1}; otherwise, it is selected from P_i. The following steps show the procedure for creating the P_{IV} position:

$j = 1$
$While\ j \leq \dim(P)\ do$:
    $if\ rand > threshold$:
        $P_{IV}[j] = P_{i+1}[j]$
    $else$:
        $P_{IV}[j] = P_i[j]$
    $j = j + 1$

After creating the information vector, P_{IV}, it will be evaluated using the objective function, and if its evaluation is better than both P_i and P_{i+1}, then P_{IV} will replace P_i to be used in the next iteration of the algorithm. The full steps of the Modified Emperor Penguin Algorithm Optimizer (MEPO) can be shown as follows:

$Input\ the\ Populatin\ size\ (N), MaxItr, and\ R\ parameters$
$Generate\ the\ initial\ population$
$Evaluate\ each\ solution\ in\ population\ and\ store\ the\ best\ s$
$Itr = 1$
$While\ Itr \leq MaxItr\ do$:
    $i = 1$
    $While\ i \leq N\ do$:
        $TA = T - \frac{MaxItr}{Itr - MaxItr}$
        $A = M \times (TA + |P_{best} - P_i| \times rand) - TA$
        $S = \left(\sqrt{f \cdot e^{-Itr/l} - e^{-Itr}}\right)^2$
        $D = |S \cdot P_i - rand\ P_{best}|$
        $P_{i+1} = P_i - A \cdot D$
        $j = 1$
        $While\ j \leq \dim(P)\ do$:
            $if\ rand > threshold$:
                $P_{IV}[j] = P_{i+1}[j]$
            $else$:
                $P_{IV}[j] = P_i[j]$
            $j = j + 1$
        $if\ f(P_{i+1}) \leq f(P_{best})\ then$:
            $P_{best} = P_{i+1}$
        $i = i + 1$
    $Itr = Itr + 1$
$Return\ Gbest$

The time complexity of the algorithm is the same as the classical EPO presented by Dhiman and Kumar [4], which is $o(k \times n \times MaxItr \times d \times N)$. The next section shows the comparative results which implemented to compare the MEPO with the classical EPO and another modification of the EPO algorithm presented by Xing [12]. All of the are coded using python programming and they code is available on https://github.com/ ahmedssssA/EPOIV.

4 Comparative results

Both algorithms, EPO and MEPO, were implemented using Python programming language on a PC equipped with a Core i5 processor clocked at 3.40GHz and 4 gigabytes of RAM. Another algorithm is selected for the comparative results from the previous work listed in Table 1. The selected algorithm is the EPO that utilizes the Levy flight and gaussian mutation (EPOLG) presented by Xing [12].To evaluate the performance of the algorithms, a set of benchmark problems available at https://www.sfu .ca/~ssurjano/optimization.html are selected as shown in Table 2. The parameters of the algorithms are set as high as possible to demonstrate their effectiveness in finding the global minimum. Specifically, the population size (N) is set to 100, the maximum number of iterations ($MaxItr$) is set to 100, and R is randomly selected for each solution in each iteration from the interval $[0,\ 1]$.

The selected optimization test functions are chosen to cover functions with different shapes, including those with many local minima, bowl-shaped, plate-shaped, steep ridges/drops, and valley-shaped. Figure 2 and Figure 3 present the comparison between EPO, EPOLG, and MEPO. The figures show that the proposed MEOP algorithm has a higher efficiency in terms of convergence and solving the stagnation problem of the multi-modal functions. The functions Ackley, Bent Ciger, Easom, Michalewicz, and Schwefel show stagnation with both EPO and EPOLG algorithms, and this stagnation is solved with the MEPO algorithm. Moreover, Table 3 presents the results of the implementation for each algorithm. The optimized results are shown in bold, and all of them are associated with the MEPO algorithm. MEPO shows significant improvement in terms of mean results and the absolute relative standard deviation (RSD), which reflects the robustness of the MEPO algorithm's results.

Table 2 Selected test optimization problems

No.	Function Name	$f(x)$	Global Minimum
1	Ackley	$20\left(e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i{}^2}}\right)-\left(e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)}\right)+20+e, x_i\in[-33,33]$	$f(x^*)=0, x^*=(0,\dots,0)$
2	Bent Cigar	$x_1^2+10^6\sum_{i=2}^{n}x_i^2, x_i\in[-100,100]$	$f(x^*)=0, x^*=(0,\dots,0)$
3	Bohachevsky	$x_1{}^2+2x_2{}^2-0.3\cos(31\pi x_1)-0.4\cos(31\pi x_2)+0.7,$ $x_i\in[-100,100]$	$f(x^*)=0, x^*=(0,0)$
4	Booth	$(x_1+2x_2-7)^2+(2x_1+x_2-5)^2, \quad x_i\in[-10,10]$	$f(x^*)=0, x^*=(1,3)$
5	Bukin	$100\sqrt{\lvert x_2-0.01x_1{}^2\rvert}+0.01\lvert x_1+10\rvert, \quad x_i\in[-15,3]$	$f(x^*)=0, x^*=(-10,0)$
6	Cross-in-Tray	$-0.0001\left(\left\lvert sin(x_1)\,sin(x_2)e^{\left\lvert 100-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert}\right\rvert+1\right)^{0.1}, \quad x_i\in[-15,15]$	$f(x^*)=-2.06261, x^*=(1.3491,-1.3491)$
7	Drop Wave	$-\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5(x_1^2+x_2^2)+2}, x_i\in[-5.12,5.12]$	$f(x^*)=-1.5, x^*=(0,0)$
8	Discus	$10^6x_1^2+\sum_{i=2}^{D}x_i^2, x_i\in[0,100]$	$f(x^*)=0, x^*=(0,\dots,0)$
9	Easom	$-\cos(x_1)\cos(x_2)\,e^{\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)}, \quad x_i\in[-100,100]$	$f(x^*)=-1, x^*=(\pi,\pi)$
10	Eggholder	$-(x_2+47)\sin\left(\sqrt{\left\lvert x_2+\frac{x_1}{2}+47\right\rvert}\right)-x_1\sin\left(\sqrt{\lvert x_1-(x_2+47)\rvert}\right), x\in[-500,500]$	$f(x^*)=-959.6407, x^*=(512,404.2319)$
11	Griewank	$\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1, x\in[-600,600]$	$f(x^*)=0, x^*=(0,\dots,0)$
12	Holder Table	$-\left\lvert\sin(x_1)\cos(x_2)e^{\left(\left\lvert 1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)}\right\rvert, x_i\in[-10,10]$	$f(x^*)=-19.2085, x^*=(8.05502,-9.6645$
13	Michalewicz	$-\sum_{i=1}^{d}\sin(x_i)\sin^{20}\left(\frac{ix_i^2}{\pi}\right), x\in[0,\pi]$	$f(x^*)=-1.8013, x^*=(2.20,1.57)$
14	Modified Schwefel	$418.9829\times D-\sum_{i=1}^{D}g(z_i), z_i=x_i+4.209687462275036\mathrm{e}+002, x_i\in[-500,500]$ $g(z_i)=\begin{cases}z_i\sin\left(\lvert z\rvert^{\frac{1}{2}}\right), & if\ \lvert z_i\rvert\le 500\\ (500-mod(z_i,500))\sin\left(\sqrt{500-\lvert mod(z_i,500)\rvert}\right)-\frac{(z_i-500)^2}{1000D}, & if\ z_i\\ (mod(\lvert z_i\rvert,500)-500)\sin\left(\sqrt{\lvert mod(\lvert z_i\rvert,500)-500\rvert}\right)-\frac{(z_i+500)}{1000D}, & if\end{cases}$	$f(x^*)=0, x^*=[0,\dots,0]$
15	Rastrigin	$10d+\sum_{i=1}^{d}[x^2-10\cos(2\pi\,x_i)], \quad x_i\in[-5,5]$	$f(x^*)=0, x^*=[0,\dots,0]$
16	Rosenbrock	$\sum_{i=1}^{d-1}[100(x_{i+1}-x_i{}^2)^2+(x_i-1)^2], \quad x_i\in[-5,10]$	$f(x^*)=0, x^*=(1,\dots,1)$

No.	Function Name	$f(x)$	Global Minimum
17	Schwefel	$418.9829d - \sum_{i=1}^{d} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right), \quad x_i \in [-500, 500]$	$f(x^*) = 0, x^* = (420.9687, \ldots, 420.9$
18	six-hump	$\left(4 - 2.1{x_1}^2 + \frac{{x_1}^4}{3}\right){x_1}^2 + x_1x_2 + (-4 + 4{x_2}^2){x_2}^2, \quad x_i \in [-3, 3]$	$f(x^*) = -1.0316, x^* = (0.0898, -0.7126)$
19	Sphere	$\sum_{i=1}^{n} {x_i}^2, x_i \in [-5, 5]$	$f(x^*) = 0, x^* = [0, \ldots, 0]$
20	Zakharov	$\sum_{i=1}^{d} {x_i}^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{d} 0.5ix_i\right)^4, \quad x_i \in [-5, 10]$	$f(x^*) = 0, x^* = (0, \ldots, 0)$

Table 3 The results of algorithms

Function	EPO Means	EPOLG Means	MEPO Means	EPO RSD	EPOLG RSD	MEPO RSD
Ackley	7.235	3.711	***0.277***	0.255	***0.228***	0.233
Bent Cigar	14.572	16.693	***0.367***	10.314	14.028	***0.907***
Bohachevsky	0.497	0.528	***0.024***	0.649	0.555	***0.142***
Booth	0.003	0.023	***0.000***	1.554	1.178	***1.131***
Bukin	1.335	1.731	***0.455***	0.655	0.424	***0.290***
Cross in tray	-2.063	-2.062	***-2.063***	0.000	0.000	***0.000***
Drop Wave	-0.989	-0.998	***-0.998***	0.024	0.033	***0.024***
Discus	20.622	17.352	***0.030***	12.782	10.151	***0.309***
Easom	-0.765	-0.449	***-1.000***	0.350	0.627	***0.001***
Eggholder	-938.908	-939.129	***-950.677***	0.012	0.011	***0.010***
Griewank	26.045	10.557	***1.259***	0.533	0.918	***0.124***
Holder Table	-19.197	-19.179	***-19.208***	0.001	0.001	***0.000***
Michalewicz	-6.149	-4.682	***-8.462***	0.101	0.076	***0.016***
Modified Schwefel	1311.743	1199.826	***1.475***	0.182	0.360	***0.129***
Rastrigin	20.361	9.611	***0.023***	0.462	0.872	***0.437***
Rosenbrock	7.525	11.891	***4.198***	1.240	1.325	***0.325***
Schwefel	1920.748	1909.122	***16.103***	0.080	0.084	***0.043***
Six hump	-1.032	-1.031	***-1.032***	0.000	0.000	***0.000***
Sphere	0.019	0.023	***0.002***	1.209	0.683	***0.262***
Zakharov	10.105	0.632	***0.506***	0.609	0.664	***0.074***

5 Conclusion

This paper presents a new improvement to the EPO algorithm presented by Dhiman and Kumar [4]. After coding the algorithm using Python programming and implementing it on a set of test optimization problems with various shapes such as many local minima, bowl-shaped, plate-shaped, steep ridges/drops, and valley-shaped, it was found that the EPO algorithm stagnated in some functions without showing any improvement. The proposed modification provides more accurate results than the original algorithm in most of the considered test functions, making the proposed algorithm more reliable than the original one. In addition, the algorithm is compared with another modification of the algorithm, EPOLG, that proposed by Xing [12] that proposed a modification in the relocating procedure of the algorithm by having levy flight and gaussian mutations. Although of his new modifications, the algorithm still stagnates on the multi-modal functions. The proposed MEPO in this paper shows a high efficiency on solving the stagnation of the multi-modal functions and shows a high robustness in comparison with the other algorithms.

Future research may include one of the following points:

- Developing further improvements to increase the efficiency of the algorithm and apply it to CEC2017 problems.
- Applying the algorithm to solve real-world problems, such as transshipment, transportation, and job shop scheduling problems.
- Optimizing the parameters of the algorithm using design of experiments.
- Hybridizing the algorithm with other metaheuristic algorithms to improve its efficiency

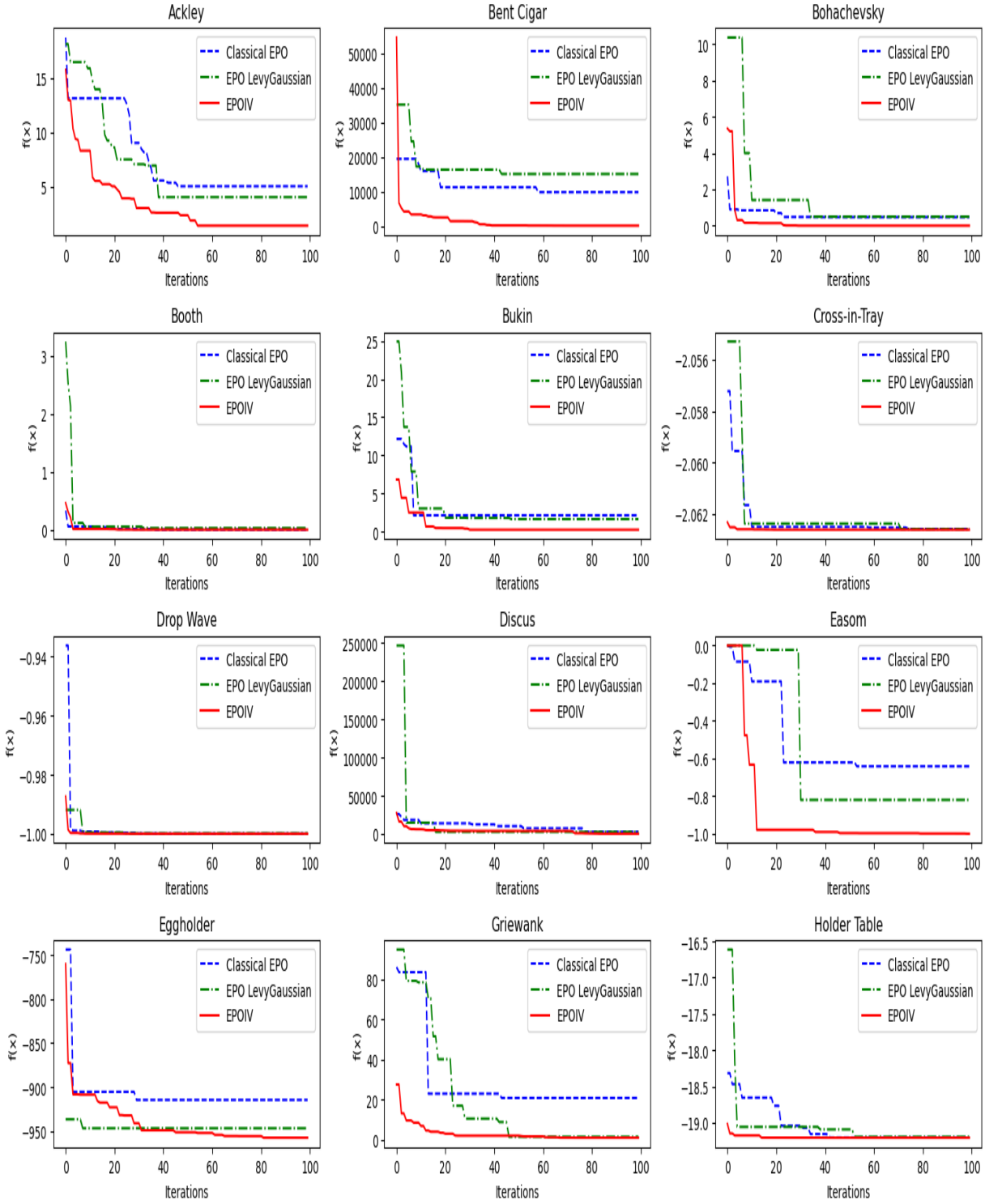

Figure 2: The results of the first 10 functions

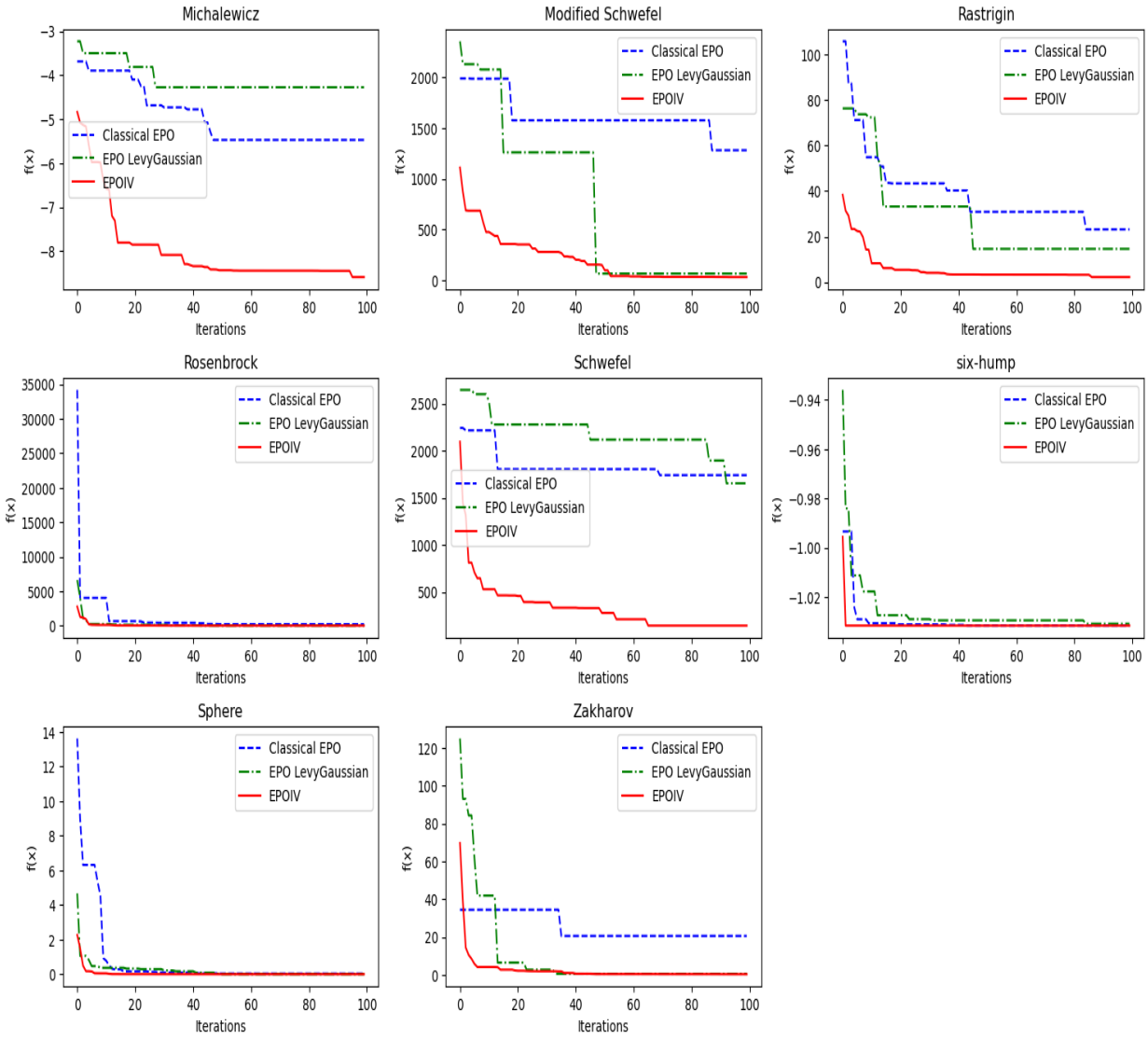

Figure 3: The results of the second 10 functions

References

[1] M. A. Kader, K. Z. Zamli, and B. S. Ahmed, "A systematic review on emperor penguin optimizer," *Neural Comput. Appl.*, vol. 33, no. 23, pp. 15933–15953, 2021, doi: 10.1007/s00521-021-06442-4.

[2] P. R. Murthy, *Operations research (linear programming)*. bohem press, 2005.

[3] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "Hybrid clustering analysis using improved krill herd algorithm," *Appl. Intell.*, vol. 48, no. 11, pp. 4047–4071, 2018, doi: 10.1007/s10489-018-1190-6.

[4] G. Dhiman and V. Kumar, "Emperor penguin optimizer: A bio-inspired algorithm for engineering problems," *Knowledge-Based Syst.*, vol. 159, pp. 20–50, 2018, doi: 10.1016/j.knosys.2018.06.001.

[5] G. Dhiman *et al.*, "BEPO: A novel binary emperor penguin optimizer for automatic feature selection," *Knowledge-Based Syst.*, vol. 211, p. 106560, 2021, doi: 10.1016/j.knosys.2020.106560.

[6] H. Kaur, A. Rai, S. S. Bhatia, and G. Dhiman, "MOEPO: A novel Multi-objective Emperor Penguin Optimizer for global optimization: Special application in ranking of cloud service providers," *Eng. Appl. Artif. Intell.*, vol. 96, p. 104008, 2020, doi: 10.1016/j.engappai.2020.104008.

[7] M. A. Sameh, M. I. Marei, M. A. Badr, and M. A. Attia, "An optimized pv control system based on the emperor penguin optimizer," *Energies*, vol. 14, no. 3, p. 751, 2021, doi: 10.3390/en14030751.

[8] J. Yang and H. Gao, "Cultural Emperor Penguin Optimizer and Its Application for Face Recognition," *Math. Probl. Eng.*, vol. 2020, 2020, doi: 10.1155/2020/9579538.

[9] H. Jia, K. Sun, W. Song, X. Peng, C. Lang, and Y. Li, "Multi-Strategy Emperor Penguin Optimizer for RGB Histogram-Based Color Satellite Image Segmentation Using Masi Entropy," *IEEE Access*, vol. 7, pp. 134448–134474, 2019, doi: 10.1109/ACCESS.2019.2942064.

[10] F. Tang, J. Li, and N. Zafetti, "Optimization of residential building envelopes using an improved Emperor Penguin Optimizer," *Eng. Comput.*, vol. 38, no. 2, pp. 1395–1407, 2022, doi: 10.1007/s00366-020-01112-w.

[11] H. Kaur, A. Rai, S. S. Bhatia, and G. Dhiman, "MOEPO: A novel Multi-objective Emperor Penguin Optimizer for global optimization: Special application in ranking of cloud service providers," *Eng. Appl. Artif. Intell.*, vol. 96, 2020, doi: 10.1016/j.engappai.2020.104008.

[12] Z. Xing, "An improved emperor penguin optimization based multilevel thresholding for color image segmentation," *Knowledge-Based Syst.*, vol. 194, 2020, doi: 10.1016/j.knosys.2020.105570.

[13] X. Lu, Y. Yang, P. Wang, Y. Fan, F. Yu, and N. Zafetti, "A new converged Emperor Penguin Optimizer for biding strategy in a day-ahead deregulated market clearing price: A case study in China," *Energy*, vol. 227, 2021, doi: 10.1016/j.energy.2021.120386.