MDPI

Article

EEGO: an extended version of Eel and grouper optimizer for global optimization problems

Glykeria Kyrou[1], Vasileios Charilogis[2] and Ioannis G. Tsoulos[3,*]

[1] Department of Informatics and Telecommunications, University of Ioannina, 47150 Kostaki Artas, Greece; g.kyrou@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, 47150 Kostaki Artas, Greece; v.charilog@uoi.gr
[3] Department of Informatics and Telecommunications, University of Ioannina, 47150 Kostaki Artas, Greece;itsoulos@uoi.gr
* Correspondence: itsoulos@uoi.gr

Abstract: The problems of finding a global minimum of a function are increasingly applied to real-world problems. As a result, a variety of computational techniques have been developed to better locate the global minimum. A decisive role is played by evolutionary techniques, which simulate natural processes and aim to find the global minimum of multidimensional functions. A recently introduced evolutionary technique is the optimal Eel and Grouper (EGO) algorithm, which is inspired by the symbiotic interaction and foraging strategy of eels and groupers in marine ecosystems. The EGO algorithm is characterized for its reliability in locating the global minimum. In this paper, modifications are proposed that aim to improve the reliability and speed of the above technique, such as the application of a termination technique based on stochastic observations and an innovative sampling method. The proposed method was tested on several problems from the relevant literature and a comparative study was made with other global optimization techniques with promising results.

Keywords: Global optimization; Meta-heuristic; Stochastic techniques; Swarm based methods

1. Introduction

The basic goal of global optimization is to find the global minimum by searching for the appropriate scope of each problem. Primarily, a global optimization method aims to discover the global minimum of a continuous function, and it is defined as

$$x^* = \arg\min_{x \in S} f(x) \quad (1)$$

with S:

$$S = [a_1, b_1] \times [a_2, b_2] \times \ldots [a_n, b_n]$$

Global optimization is an integral part of many areas of our daily lives. One of these fields is computer science[**? ? ?**]. In addition to the computer field, global optimization finds application in sciences such as mathematics[**? ? ? ?**], physics[**? ? ?**], chemistry[**? ? ?**] and medicine[**? ? ?**]. Optimization methods are divided into two categories, deterministic[**? ? ?**] and stochastic[**? ? ?**]. The first category, i.e. that of deterministic methods, has as its main objective the identification of the overall optimal solution and is mainly used in simple problems. On the contrary, stochastic methods are mainly used in complex problems.

Swarm Intelligence Algorithms[**? ? ? ? ?**] as a source of inspiration and collective behavior of insects and other animals. Intelligence Algorithms mimic systems in which agents interact locally and cooperate worldwide to solve optimization problems. Swarm intelligence algorithms are very important tools for dealing with complex optimization

Citation: Kyrou, G.; Charilogis, V.; Tsoulos, I.G. EEGO: an extended version of Eel and grouper optimizer for global optimization problems. *Journal Not Specified* **2024**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

problems in a wide range of applications[**?**]. Characteristic examples of such algorithms are the WOA algorithm **? ? ? ? ? ?** , SCA algorithm [**? ? ? ? ?**]and SSA algorithm[**? ? ? ? ?**].

The EGO algorithm is a reliable technique for real-world optimization problems. The EGO algorithm is inspired by the symbiotic interaction and foraging strategy of eels and groupers in marine ecosystems. In nature, there are complex interactions between biological species. Relationships of this type are classified into five categories[**? ?**]:1. Naturalism: Where two species can live in an ecosystem without affecting each other. 2. In predation, where one creature dies to feed another. 3. Parasitism: where one species causes harm to another without always killing it. 4. In competitive mode, the same or different organizations compete for resources. 5. Mutualism: [**? ? ?**]when two organisms have a beneficial INTERACTION. Bshary et al. [**?**]consider that target ingestion, something observed in eels and groupers, is a necessary condition for interspecific cooperative hunting to occur. Intraspecific predation could increase the hunting efficiency of predators by mammals. According to Ali Mohammadzadeh and Seyedali Mirjalili the EGO optimization algorithm [**?**]generates a set of random answers, then stores the best answers found so far, allocates them to the target point, and changes the answers with them. As the number of iterations increases, the limits of the sine function are changed to enhance the phase of finding the best solution. This method stops the process when the iteration exceeds the maximum number. Because the EGO optimization algorithm generates and boosts a collection of random responses, it has the advantage of increased local optimum discovery and avoidance compared to individual methods.

This paper introduces some modifications to the EGO algorithm in order to improve its efficiency. The proposed amendments are presented below:

- The addition of a sampling technique based on the K-means method[**? ? ? ? ? ?**]. The sampled points will help to find the global minimum of the function in the most efficient way. Additionally, by using this method, points that are in proximity can be rejected.
- The use of a termination technique based on stochastic observations. At each iteration of the algorithm, the smallest value is recorded. Once it remains stable for a predetermined number of iterations, the method terminates. The present termination method helps with the fastest termination without unnecessarily wasting computing time.
- Using mod1 and mod2 and mod3 parameters which take values of either 1, which means it is Disable or 2, which means it is Enable. mod1 specifies how randomness will be used when updating solutions. When mod1 is set to 1 p, it takes values from 0 to 1, while when mod1 is set to 2 p it takes values from -1.0 to 1.0. The mod2 parameter affects how the variables f1 and f2 are defined, which in turn affects the calculation of the new positions. The mod3 parameter determines how the algorithm handles thresholds when agent positions exceed predefined minimum and maximum thresholds. When mod3 is set to 1, it ensures that all positions remain within valid limits. When mod3 is set to 2 these positions are not taken into account and the fish wait for better positions to attack.

The rest of this paper is divided into the following sections: in section **??**, the proposed method is fully described.

2. The proposed method

2.1. The main steps of the algorithm

The main steps of the used global optimization method are the following:

1. **Initialization step**.
   (a) Define as N_c the number of elements in the search Agents
   (b) Define as N_g, the maximum number of allowed iterations.
   (c) Initialize randomly the members of the search Agents in set S.
2. **Calculation step**.
   (a) While $[(t \prec Ng)]$

(b) Update Variables a and starrvation_rate where starrvation_rate is a number in [0,100]. According to types:

- $a = 2 - 2 * (t \ / \ Ng)$
- $(starvatiorate = 100 * (t / \ Ng))$

(c) Compute the fitness of each search Agents.

(d) Sort all solutions in the current population fro the best to worst according to the function value.

For $j = 1, \ldots, N_c$ **do**

- Update variables r1, r2, r3, r4, C1, C2 and p : where r_1, r_2 are random numbers in , $[0, 1]$, C1 is a rondom number in [-a,a], C2 is a random number in [0,2]. According to types:
  - $r3 = (a - 2) * r1 + 2$
  - $r4 = 100 * r2$
  - $C1 = 2 * a * r1 - a$
  - $C2 = 2 * r1$
  - $b = a * r2$
- Update the position of each search Agent:
- $if(r4 <= starvation_rate)$
- if mod1 has the value 1
  - p is a random number in $[0.0, 1.0]$
- if mod1 has the value 2
  - p is a random number in $[-1.0, 1.0]$
- if mod2 has the value 1
  - $f1 = 0.8$
  - $f2 = 0.2$
- if mod2 has the value 2
  - f1 is a random number in $[0.0, 2.0]$
  - f1 is a random number in $[-2.0, 0.0]$

If $(p \prec 0.5)$

- Change the location of current search by :$[(f1 * X1 \dot{+} f2 * X2) \ /2]$

else if $(p \geq 0.5)$

- Change the location of current search by :$(f2 * X1 \dot{+} f1 * X2 \ /2)$

End if

End For

(e) Ensure No Agents leave the search area

(f) Evaluate the performance of each search Agents

(g) Update XPrey

- if there is a better solution

(h) Set $t = t + 1$

(i) End while

(j) Return the best solution XPrey

3. Termination check step

(a) Set $t = t + 1$

(b) Calculate to stopping rule in the work of Charilogis [**?**]

(c) If the termination criteria are not met then go to Calculation step, **else** terminate.

2.2. The proposed sampling procedure

The sampling technique applied in this work initially generates samples from the objective function. Then, through the K-means method, only the recognized centers are selected as final samples. This technique, which is an achievement of James MacQueen [41], is one of the most well-known clustering algorithms in the broad research community, both in data analysis and in machine learning [42, 43] and pattern recognition[**?**]. The algorithm mainly aims to estimate the centers of possible groups from a set of samples. Next, the basic steps of the algorithm are presented:

1. **Define** as k the number of clusters.
2. Randomly select N_m initial points $x_i,\ i = 1, \ldots, N_m$ from the objective function.
3. Randomly assign each point $x_i,\ i = 1, ..., N_m$ in a cluster $S_j,\ j = 1, \ldots, k$.
4. **For** every cluster $j = 1..k$ **do**
   - **Set** as M_j the number of points in S_j
   - **Calculate** the center of the cluster c_j as

$$c_j = \frac{1}{M_j} \sum_{x_i \in S_j} x_i$$

5. **End For**
6. **Repeat the following steps:**
   - Set $S_j = \{\},\ j = 1..k$
   - **For** each point $x_i,\ i = 1, ..., N_m$ **do**
     - **Set** $j^* = \text{argmin}_{m=1}^{k}\{D(x_i, c_m)\}$. The function $D(x, y)$ is the Euclidean distance of points (x, y).
     - **Set** $S_{j^*} = S_{j^*} \cup \{x_i\}$.
   - **End For**
   - **For** each center $c_j,\ j = 1..k$ **do**
     - **Update** the center c_j as

$$c_j = \frac{1}{M_j} \sum_{x_i \in S_j} x_i$$

   - **End For**
7. **If** there is no significant change in centers c_j **terminate** the algorithm and return the k centers as the final set of samples.

3. Results

This section will begin with a detailed description of the functions that will be used in the experiments, followed by an analysis of the experiments performed and comparisons with other global optimization techniques.

3.1. Test functions

The functions used in the experiments have been proposed in a series of relative works[**? ?**] and they cover various scientific fields, such as medicine, physics, engineering, etc. Also, these objective functions have been used by many researchers in a variety of publications.[**? ? ? ? ?**] The definitions of these functions are given below:

- **Bf1** (Bohachevsky 1) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$$

- **Bf2** (Bohachevsky 2) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$$

- **Bf3** (Bohachevsky 3) function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1 + 4\pi x_2) + \frac{3}{10}$$

- **Branin** function:

$$f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$$

with $-5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15$.

- **Camel** function:

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5,5]^2$$

- **Easom** function:

$$f(x) = -\cos(x_1)\cos(x_2)\exp\Big((x_2-\pi)^2 - (x_1-\pi)^2\Big)$$

with $x \in [-100,100]^2$.

- **Exponential** function, defined as:

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \leq x_i \leq 1$$

In the conducted experiments the values $n = 4, 8, 16, 32$ were used.

- **Griewank2** function:

$$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}, \quad x \in [-100,100]^2$$

- **Griewank10** function. The function is given by the equation

$$f(x) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

with $n = 10$.

- **Gkls** function[**?**]. $f(x) = \text{Gkls}(x, n, w)$, is a constructed function with w local minima presented in [**?**], with $x \in [-1,1]^n$. For the conducted experiments the values $n = 2, 3$ and $w = 50$ were utilized.

- **Goldstein and Price function**

$$\begin{aligned} f(x) \quad = \quad & \Big[1 + (x_1 + x_2 + 1)^2 \\ & \Big(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\Big)\Big] \times \\ & [30 + (2x_1 - 3x_2)^2 \\ & \Big(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\Big)\Big] \end{aligned}$$

With $x \in [-2,2]^2$.

- **Hansen** function: $f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i] \sum_{j=1}^{5} j\cos[(j+1)x_2 + j]$, $x \in [-10,10]^2$.
- **Hartman 3** function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$$

with $x \in [0,1]^3$ and $a = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}$$

- **Hartman 6** function:

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$$

with $x \in [0,1]^6$ and $a = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

- **Potential** function, this function stands for the energy of a molecular conformation of N atoms, that interacts using via the Lennard-Jones potential[**?**]. The function is defined as:

$$V_{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

For the conducted experiments the values $N = 3,\ 5$ were used.

- **Rastrigin** function.

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1,1]^2$$

- ***Rosenbrock*** *function*.

$$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \le x_i \le 30.$$

The values $n = 4,\ 8,\ 16$ were used in the conducted experiments.

- **Shekel 5** function.

$$f(x) = -\sum_{i=1}^{5} \frac{1}{(x - a_i)(x - a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$

- **Shekel 7** function.

$$f(x) = -\sum_{i=1}^{7} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}$.

- **Shekel 10** function.

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$.

- **Sinusoidal** function defined as:

$$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right), \quad 0 \le x_i \le \pi.$$

The values of $n = 4, 8, 16$ were used in the conducted experiments.

- **Test2N** function:

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5, 5].$$

For the conducted experiments the values $n = 4, 5, 6, 7$ were used.

- **Test30N** function:

$$f(x) = \frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\Big((x_i-1)^2\Big(1+\sin^2(3\pi x_{i+1})\Big)\Big) + (x_n-1)^2\Big(1+\sin^2(2\pi x_n)\Big)$$

The values $n = 3, 4$ were used in the conducted experiments.

3.2. Experimental results

The following applies to table **??**:

- The column FUNCTION denotes the name of the objective problem.