New Evolutionary Selection Operators for Snake Optimizer

Ruba Abu Khurma[1], Moutaz Alazab[2], J. J. Merelo[3] and Pedro A. Castillo[3]
[1]*Department of Computer Science, Al-Ahliyya Amman University, Amman, Jordan*
[2]*Department of Artificial Intelligence, Al-Balqa University, Al-Salt, Jordan*
[3]*Department of Computer Architecture and Computer Technology, ETSIIT and CITIC, University of Granada, Granada, Spain*

Keywords: Snake Optimizer, Evolutionary Operators, Selection Schemes.

Abstract: Evolutionary algorithms (EA) adopt a Darwinian theory which is known as "survival of the fittest". Snake Optimizer (SO) is a recently developed swarm algorithm that inherits the selection principle in its structure. This is applied by selecting the fittest solutions and using them in deriving new solutions for the next iterations of the algorithm. However, this makes the algorithm biased towards the highly fitted solutions in the search space, which affects the diversity of the SO algorithm. This paper proposes new selection operators to be integrated with the SO algorithm and replaces the global best operator. Four SO variations are investigated by individually integrating four different selection operators: SO-roulettewheel, SO-tournament, SO-linearrank, and SO-exponentialrank. The performance of the proposed SO variations is evaluated. The experiments show that the selection operators have a great influence on the performance of the SO algorithm. Finally, a parameter analysis of the SO variations is investigated.

1 INTRODUCTION

Natural systems live in groups characterized by decentralization and self-organization among their members (Khurma et al., 2020). These swarm systems have their special relationships between swarm members and their environment that control their search for food and sustenance. Swarm Intelligence researchers investigate the collective behavior of animals and turn their social relationships into mathematical methodologies (Abu Khurma et al., 2022).

The SO algorithm is a recently developed swarm algorithm by Fatma (Hashim and Hussien, 2022). The main inspiration for the SO algorithm comes from the behavior of snakes in nature. Many environmental factors influence the behavior of snakes. For example, snake mating occurs when the temperature is cold and there is food. Otherwise, snakes are looking for food. The SO methodology translates this inspiration into two phases of exploration and exploitation. Exploration is the situation where no food is found and the temperature is cold so that the snakes (solutions) search globally in the search space. The stage of exploitation includes several cases. If the food is available and the temperature is hot, the snakes eat the food that is present. If food is available and the temperature is cool, snakes enter the mating process. The mating process has two modes, either fighting mode or mating mode. The fighting mode makes the snakes fight until the male gets the best female and the female gets the best male. Mating mode between a pair of snakes occurs depending on the amount of food. Mating may result in the birth of new snakes.

The SO algorithm evaluates solutions at each iteration to get the best solutions in the male group and female group which are called $Snakemale_{best}$ and $Snakefemale_{best}$ respectively. The SO update procedure for other solutions is guided by the positions of the best solutions. During the iterations of the algorithm, the re-position of solutions in the search space depends on the distance from the best solutions. This means that the search process is biased toward the best solution. Changing the positions of solutions concerning one point during the search affects the diversity of solutions and the exploration of the algorithm. This may lead also to premature convergence and stagnation in local minima.

In the literature, there have been vast studies that investigate the effects of evolutionary selection operators on the performance of swarm intelligent algorithms in different applications. A previous study (Khurma et al., 2021), integrated different selection

Khurma, R., Alazab, M., Merelo, J. and Castillo, P.
New Evolutionary Selection Operators for Snake Optimizer.
DOI: 10.5220/0011524300003332
In *Proceedings of the 14th International Joint Conference on Computational Intelligence (IJCCI 2022)*, pages 82-90
ISBN: 978-989-758-611-8; ISSN: 2184-2825

operators with the Moth Flame Optimizer (MFO) and use them with Levy flight operator to enhance the feature selection process. The results showed improved performance in the diagnosis of disease. The authors in (Al-Betar et al., 2018a), improved the original greedy selection operator of the Grey Wolf Optimizer (GWO) by using other less-biased selection operators extracted from evolutionary algorithms. Six GWO variations are produced and studied on 23 mathematical benchmark functions. The results show that TGWO achieved the best results. In (Al-Betar et al., 2012), the authors used new selection operators with the Harmony Search (HS) algorithm in memory consideration to replace the original random selection The results on benchmark functions showed that the selection operators affected the performance of the HS algorithm. The authors in (Al-Betar et al., 2018b), studied the effect of the selection operators on the Bat algorithm. They replaced the global-best selection with other evolutionary operators. Their evaluation on 25 IEEE-CEC2005 functions showed competitive results. In (Awadallah et al., 2019), The authors studied the effect of different selection operators on the Artificial Bee Colony (ABC) algorithm. The evaluation results on benchmark functions showed the effects of the selection operators on ABC algorithm. In (Shehab et al., 2016), the authors studied the effect of replacing the global best selection mechanism of the Particle Swarm Optimization (PSO) algorithm with other selection operators. The results showed a direct effect of the selection operators on the performance of the PSO. Bottom-line, the literature has many research papers that investigates the effect of different selection schemes on the performance of several swarm intelligent algorithms. According to the No-Free-Lunch theorem, (Adam et al., 2019), no swarm algorithm has the same performance in tackling all optimization problems. Therefore, there is always a chance for researchers to suggest new algorithms and experiment them with different optimization problems.

In this paper, new evolutionary selection operators are borrowed from the Genetic Algorithm (GA) and integrated with the SO algorithm. These operators are Roulette wheel, tournament, linear rank, and exponential rank. Each operator is integrated individually with the SO algorithm and substitutes the global best standard operator. Four SO variations are produced using these operators: SO-roulettewheel, SO-tournament, SO-linearrank, and SO-exponentialrank. The performance of each SO variation is investigated and evaluated using the standard benchmark mathematical functions.

The remaining parts of this paper are organized as follows: Section 2 presents the SO algorithm. Section 3 discusses the proposed selection operators integrated with SO. Section 4 analyzes and discusses the experimental results. Finally, Section 5 summarizes the paper and determines some future works.

2 SNAKE OPTIMIZER (SO)

This section presents the mathematical model of a recently published SO algorithm (Hashim and Hussien, 2022). The following points explains in detail SO steps:

- Initializing solutions: SO starts by initiating a set of random solutions in the search space using Eq.(1). These solutions compose the snakes population to be optimized in the next steps.

$$Snake_i = Snake_{min} + random \times (Snake_{max} - Snake_{min}) \quad (1)$$

where $Snake_i$ is the location in the search space of the i_{th} solution in the swarm. $random$ is a random number $\in [0,1]$. $Snake_{max}$ and $Snake_{min}$ are the minimum and the maximum values respectively for the studied problem.

- Division of solutions: the population is divided into two parts (50% male and 50% female) using Eq. (2) and Eq. (3)

$$Num_{male} \approx Num/2 \quad (2)$$

$$Num_{female} \approx Num - Num_{male} \quad (3)$$

where Num is the size of the population (all snakes). Num_{male} is the number of the male solutions. Num_{female} is the number of female solutions.

- Evaluate solutions: get the best solution from the male group ($Snakebest_{male}$), female group ($Snakebest_{female}$) and find the location of the food L_{food}. Two other concepts are defined which are the temperature ($Temperature$) and the quantity of food ($Qantity$) as in Eq.(4) and Eq.(5) respectively.

$$Temperature = Exp(\frac{-Curiter}{Totiter}) \quad (4)$$

where $Curiter$ is the current iteration and $Totiter$ is the number of all iterations.

$$Qantity = Const_1 \times Exp(\frac{Curiter - Totiter}{Totiter}) \quad (5)$$

where $Const_1$ is a constant value equal 0.5.

- Exploring the search space (food is not found): this depends on using a specified threshold value. If $Quantity < 0.25$, the solutions search globally by updating their locations with respect to a specified random location in the search space. This modeled by Eq.(6)-Eq.(9)

$$Snakemale_i(iter+1) = Snakemale_{rand}(iter) \pm Const_2 \times ABmale \times ((Snake_{max} - Snake_{min}) \times rand + Snake_{min}) \quad (6)$$

where $Snakemale_i$ is i_{th} male solution, $Snakemale_{rand}$ is the location of random male solution, $rand$ is a random number $\in [0,1]$ and AB_{male} is the ability of the male solution to find the food and can be computed using Eq.(7):

$$ABmale = Exp(-\frac{Fitnessmale_{rand}}{Fitnessmale_i}) \quad (7)$$

where $Fitnessmale_{rand}$ is the fitness of $Snakemale_{rand}$ and $Fitnessmale_i$ is the fitness of i_{th} solution the in male group and $Const_2$ is a constant equals 0.05.

$$Snakefemale_i(iter+1) = Snakefemale_{rand}(iter) \pm Const_2 \times ABfemale \times ((Snake_{max} - Snake_{min}) \times rand + Snake_{min}) \quad (8)$$

where $Snakefemale_i$ is i_{th} female solution, $Snakefemale_{rand}$ is the location of random female solution, $rand$ is a random number $\in [0,1]$ and AB_{female} is the ability of the female solution to find the food and can be computed using Eq.(9):

$$ABfemale = Exp(-\frac{Fitnessfemale_{rand}}{Fitnessfemale_i}) \quad (9)$$

where $Fitnessfemale_{rand}$ is the fitness of $Snakefemale_{rand}$ and $Fitnessfemale_i$ is the fitness of i_{th} solution the in male group and $Const_2$ is a constant equals 0.05.

- Exploiting the search space (Food is found) If the quantity of food is greater than a specified threshold $Quantity > 0.25$ then the temperature is checked. If $Temperature > 0.6$ (hot), The solutions will move to the food only.

$$Snake_{(i,j)}(iter+1) = L_{food} \pm Const_3 \times Temperature \times rand \times (L_{food} - Snake_{(i,j)}(iter)) \quad (10)$$

where $Snake_{(i,j)}$ is the location of a solution (male or female), L_{food} is the location of the best solutions, and $Const_3$ is constant value and equals 2.

If $Temperature > 0.6$ (cold), The snake will be in the fight mode or mating mode Fight Mode.

$$Snakemale_i(iter+1) = Snakemale_i(iter) \pm Const_3 \times FAMtimesrand \times (Snakefemale_{best} - Snakemale_i(iter)) \quad (11)$$

where $Snakemale_i$ is the i_{th} male location, $Snakefemale_{best}$ is the location of the best solution in female group, and FAM is the fighting ability of male solution.

$$Snakefemale_i(itert+1) = Snakefemale_i(iter+1) \pm Const3 \times FAF \times rand \times (Snakemale_{best} - Snakefemale_i(iter+1)) \quad (12)$$

where $Snakefemale_i$ is the i_{th} female location, $Snakemale_{best}$ is the location of the best solution in the male group, and FAF is the fighting ability of the female solution.

FAM and FAF can be computed from the following equations:

$$FAM = Exp(-\frac{Fitnessfemale_{best}}{Fitness_i}) \quad (13)$$

$$FAF = Exp(-\frac{Fitnessmale_{best}}{Fitness_i}) \quad (14)$$

where $Fitnessfemale_{best}$ is the fitness of the best solution of the female group, $Fitnessmale_{best}$ is the fitness of the best solution of male group, and $Fitness_i$ is the solution fitness.

Mating mode.

$$Snakemale_i(iter+1) = Snakemale_i(iter) \pm Const_3 \times MAm \times rand \times (Quantity \times Snakefemale_i(iter) - Snakemale_i(iter)) \quad (15)$$

$$Snakefemale_i(iter+1) = Snakefemale_i(iter) \pm Const_3 \times MAfm \times rand \times (Quantity \times Snakemale_i(iter) - Snakefemale_i(iter)) \quad (16)$$

where $Snakefemale_i$ is the location of the i_{th} solution in female group and $Snakemale_i$ is the location of the i_{th} solution in male group and MAm and MAf are the ability of male and female for mating respectively and they can be computed as follow:

$$MAm = Exp(-\frac{Fitnessfemale_i}{Fitnessmale_i}) \quad (17)$$

$$MAf = Exp(-\frac{Fitnessmale_i}{Fitnessfemale_i}) \quad (18)$$

If Egg hatch, select worst male solution and worst female solution and replace them

$$Snakemale_{worst} = Snake_{min} + rand \times (Snake_{max} - Snake_{min}) \quad (19)$$

$$Snakefemale_{worst} = Snake_{min} + rand \times (Snake_{max} - Snake_{min}) \quad (20)$$

where $Snakemale_{worst}$ is the worst solution in the male group, $Snakefemale_{worst}$ is the worst solution in female group. The diversity factor operator $\pm$ gives chance to increase or decrease locations' solution to give high probability to change the the locations of solutions in the search space in all possible directions.

Algorithm 1: Continuous SO Algorithm.

Input: Dim, UB, LB, Num, Totiter, and Curiter
Output:Best Snake
Initialize the Snakes randomly
while $Curiter \leq Totiter$ **do**
  Evaluate each Snake in groups Num_{male} and Num_{female}
  Find best male $Snakemale_{best}$
  Find best female $Snakefemale_{best}$
  Define $Tempreture$ using Eq. (4).
  Define food Quantity $Quantity$ using Eq. (5).
  if ($Quantity < 0.25$) **then**
    Perform exploration using Eq. (6) and Eq. (8)
  else if ($Temperature > 0.6$) **then**
    Perform exploitation Eq. (10)
  else if ($rand > 0.6$) **then**
    Snakes in Fight Mode Eq. (11) and Eq. (12)
  else
    Snakes in Mating Mode Eq. (15) and Eq. (16)
    Change the worst male Snake and female Snake Eq. (19) and Eq. (20)
  end if
end while
Return best Snake

3 THE PROPOSED SELECTION OPERATORS

3.1 Roulette Wheel Selection

This method selects solutions based on their fitness value in proportion to the fitness of other solutions in the population. Thus, the probability to select a solution depends on the absolute value of its fitness in relation to the fitness of other solutions in the population. The selection probability $Prob_i$ for the solution i is computed by Eq 21. In Algorithm 2, $Random$ is a random value from a uniform random distribution $U(0,1)$; $Totalprob$ is the accumulative selection probabilities of solution S^j which is defined by $Totalprob = \sum_{i=1}^{j} Prob_i$.

$$Prob_i = \frac{f(S^i)}{\sum_{j=1}^{popsize} f(S^j)} \quad (21)$$

Algorithm 2: Pseudocode for the Roulette wheel selection.

Set $Random \sim U(0,1)$
Set $F = False$
Set $Totalprob = 0$
Set $L = 0$
while $i \leq popsize$ and not (F) **do**
  $Totalprob = Totalprob + Prob_i$
  if $Totalprob > Random$ **then**
    $L = i$
    $F = True$
  end if
  $i = i + 1$
end while

3.2 Tournament Selection

Algorithm 3 shows the tournament selection method, which begins by selecting a set of random solutions from the population. The number of selected solutions is called the tournament size. The procedure proceeds by selecting the best solution in tournament T. This process is repeated n times.

Algorithm 3: Pseudocode for the Tournament selection.

Select L random solutions from the population.
Select the first-best solution from tournament with probability $Prob$.
Select the second-best solution with probability $Prob \times (1 - Prob)$.
Select the third-best solution with probability $Prob \times ((1 - Prob)^2$.
And so on...

3.3 Linear Rank Selection

This selection method overcomes the shortcomings of the Roulette wheel method by adopting the rank of solutions instead of their fitness values. Eq 22 shows that the probability of selection for a solution depends on its rank. The highest rank n is given to the best solution while the lowest rank 1 is given for the worst

solution. Once all solutions in the swarm are ranked, Algorithm 4 is applied.

$$Prob_i = \frac{rank_i}{n \times (n-1)} \tag{22}$$

Algorithm 4: Pseudocode for the Linear rank selection.

```
Set X_0 = 0
  for i = 1 to popsize do
    X_i = X_{i-1} + Prob_i
  end for
  for i = 1 to popsize do
    Generate a Random ∈ [0, popsize]
    for 1 ≤ j ≤ popsize do
      if Prob_j ≤ Random then
        Select the j^th solution
      end if
    end for
  end for
```

3.4 Exponential Rank Selection

This method sorts the ranked solutions depending on their probabilities by using exponentially weighted as in Eq 23. the value of X is between 0 and 1. If $X = 1$, the difference in the selection probability between the best and the worst solutions is ignored. If $X = 0$, this continuously increases the difference in the selection probability in such a way it produces an exponential curve along the ranked solution. Algorithm 5 shows the steps of exponential ranking, it differs from the linear ranking in the computation of the selection probabilities.

$$Prob_i = \frac{M^{rank_i}}{\sum_{j=1}^{popsize} M^{rank_j}} \tag{23}$$

Algorithm 5: Pseudocode for the Exponential rank selection.

```
Set X_0 = 0
  for i = 1 to popsize do
    X_i = X_{i-1} + Prob_i
  end for
  for i = 1 to popsize do
    Generate a Random ∈ M
    for 1 ≤ j ≤ popsize do
      if Prob_j < r then
        Select the j^th solution
      end if
    end for
  end for
```

4 ANALYSIS AND DISCUSSIONS OF THE EXPERIMENTAL RESULTS

This section perform the experiments on the four generated SO variations listed below to evaluate the effect of different selection operators on the performance of the SO algorithm. Each variation adopts a specific selection operator that is integrated with the SO algorithm:

1. SO-Global-best: it uses the SO algorithm with Global-best selection operator.
2. SO-Roulettewheel: it uses the SO algorithm with the Roulette wheel selection operator.
3. SO-Tournament: It uses the SO algorithm with the tournament selection operator.
4. SO-Linearrank: it uses the SO algorithm with the linear rank selection operator.
5. SO-Exponentialrank: it uses the SO algorithm with the exponential rank selection operator.

All the experiments are conducted using a computer with processor 11th Gen Intel(R) Core(TM) 16-1135G7@2.40GHz with 16 GB of RAM and 64-bit for Microsoft Windows 10 Pro. The MATLAB (R2010a) is used to implement the source code.

Different parameter settings are used to evaluate the SO with different selection operators. The investigated parameters are the number of dimensions and population size for each SO version as follows: number of dimensions Dim = (10, 20, and 30) (Trelea, 2003), population size = (30, 50, and 80) (Richards and Ventura, 2003). Each run is iterated 100,000 times.

The optimal parameter setting for each version is shown in Table 1. The experiments are then performed using five scenarios with different parameter settings, as shown in Table 1. Each scenario studies the capability of the two parameters, and each of these parameters includes a set of values. For example the first scenario shows the SO-Global-best with its optimal value of each parameter, as shown Dim=30, and population size= 50, These values are identified as an optimal value for the experiments, and so on for all scenarios.

More function evaluations are needed when the the number of dimensions increases. At the same time, increasing the computations increases the algorithm's reliability. The most important issue of this work is to balance between reliability and cost. Therefore, the optimal value for the number of dimensions should be between 10 and 30 and should not be

Table 1: Parameters scenarios.

Scenario Number	Selection operator	Dim	Population size
Scenario1	Global-best	30	50
Scenario2	SO-Roulettewheel	30	50
Scenario3	SO-Tournament	30	30
Scenario4	SO-Linearrank	30	30
Scenario5	SO-Exponentialrank	30	50

more than 30 when the problem becomes more complex. The results achieved in this work are consistent with (Jin et al., 2013).

For the population size parameter, assigning a value of 50 is recommended when dealing with high dimensional problems. However, selecting a population size of [30, 50] is recommended for lower dimensional problems. The values of population size are consistent with previous studies (Li-Ping et al., 2005).

This study uses 14 global benchmark functions that include both unimodal and multimodal functions. These are used commonly to solve minimization optimization problems (Civicioglu and Besdok, 2013). The purpose of adopting these functions is to evaluate the performance of the SO algorithm.

Table 2 and Table 3 show the optimal solutions obtained by the SO variations using the 14 benchmark functions. The target form using benchmark functions is to get the minimum solution and this depend on each benchmark. For most of a benchmark the best value is near Zero. However, the best value for other benchmark is near (- 450) like shifted benchmark functions. All selection operators try to be near to the best solution, but SO-Tournament obtained the first rank. The SO-Exponentialrank got the worst solution, SO-Roulettewheel, SO-Global-best and SO-Linearrank are respectively among them.

Table 2 and Table 3 summarize the results of the SO variations using the 14 benchmark functions in each scenario, as shown in Table 1. The results in Tables 2 and 3 are arranged from scenario1 to scenario5 to save the best value for each parameter, which means in scenario5 each of the selection schemes has the best values of parameters. Each SO version implemented 30 runs, and the values in the table refer to the average and standard deviations (within the parentheses). The optimal solutions appear in bold font. The results show that SO-Tournament obtains the optimal results for all the benchmark functions. SO-Global-best and SO-Roulettewheel obtaine the eight best results for the Sphere, Schwefel problem 2.22, Step, Rosenbrock, Rotated hyper-ellipsoid, Rastrigin, Ackley, and Griewank benchmark functions. SO-Linearrank obtains the best results for most of the benchmark functions. By contrast, SO-Exponentialrank obtains poor results when compared with the other selection operators, especially for the Rotated hyper-ellipsoid, Rastrigin, Shifted Sphere, and Shifted Rosenbrock benchmark functions.

5 CONCLUSION AND FUTURE WORK

This study investigates the effect of integrating different evolutionary selection operators in the structure of the SO optimizer. The work is done by replacing the original global-best solution scheme by four other selection operators. The integration of selection schemes with SO produces four variations of the SO algorithm namely SO-Roulettewheel, SO-Tournament, SO-Linearrank and SO-Exponentialrank. The SO variations aim is to apply the survival of the fittest selection principle in the search space. The experiments were performed on the global mathematical benchmark functions. The results proved that integrating the selection operators in the search space of the SO algorithm is capable to improve the balance between the global and local search phases and to alleviate the premature convergence due to entrapment in local minima. Furthermore, the SO-Tournament achieved the best results, followed by SO-Roulettewheel and SO-Globalbest, whose results were very close. SO-Linearrank and SO-Exponentialrank come in the fourth and the last place respectively. This study investigates the effect of selection schemes for the first time on the SO algorithm. For future, we intend to explore this study by considering the search time. Also, we intend to apply the SO with selection operators in specific domains to solve some real-life problems.

ACKNOWLEDGMENTS

This work is supported by the Ministerio español de Economía y Competitividad under project PID2020-115570GB-C22 (DemocratAI::UGR).

Table 2: The Average and Standard Deviation Results of Benchmark Functions.

Benchmark Function	Selection Technique	Scenario1	Scenario2	Scenario3	Scenario4	Scenario5
Sphere	SO-Global-best	5.86E-07 (8.88E-07)	3.16E-07 (1.10E-08)	0.36E-07 (0.81E-07)	2.00E-07 (9.66E-09)	**1.70E-08 (8.74E-09)**
	SO-Roulettewheel	8.40E-07 (3.06E-07)	7.12E-07 (1.34E-09)	5.47E-07 (3.14E-07)	**3.76E-10 (0.872E-11)**	1.40E-07 (3.06E-07)
	SO-Tournament	0.00E+00 (0.00E+00)	5.82E-17 (1.10E-15)	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
	SO-Linearrank	2.27E+05 (8.37E+04)	3.14E-05 (2.40E-02)	3.70E-04 (6.20E-04)	8.76E-08 (7.07E-08)	**4.46E-08 (0.12E-09)**
	SO-Exponentialrank	2.14E-02 (0.08E-02)	2.13E-03 (1.00E-02)	0.45E-03 (0.77E-02)	1.32E-04 (0.01E-04)	**0.36E-07 (0.81E-07)**
Schwefel's problem 2.22	SO-Global-best	0.0026 (0.0036)	0.0025 (0.0042)	6.88E-05 (5.54E-04)	6.77E-07 (1.32E-06)	**7.32E-08 (0.01E-10)**
	SO-Roulettewheel	0.0001 (0.0001)	4.74E-07 (0.0001)	3.65E-05 (2.66E-05)	1.33E-08 (7.64E-07)	**8.12E-10 (3.54E-09)**
	SO-Tournament	3.22E-26 (0.00E+00)	3.46E-394 (0.00E+00)	1.46E-394 (0.00E+00)	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)
	SO-Linearrank	0.0000 (0.0731)	6.24E-03 (5.73E-03)	8.65E-04 (4.80E-03)	0.01E-06 (7.72E-05)	**4.12E-07 (2.76E-07)**
	SO-Exponentialrank	1.07E+03 (2.58 E+02)	1.06E+03 (2.70E+02)	0.003E+03 (3.148E+03)	**0.81E+02 (2.76E+03)**	0.56E+03 (0.43E+03)
Step	SO-Global-best	6.17E-06 (5.40E-06)	8.40E-07 (0.10E-06)	5.10E-07 (2.46E-05)	7.83E-07 (1.44E-08)	**0.64E-09 (1.43E-08)**
	SO-Roulettewheel	0.40E-07 (1.05E-07)	0.40E-07 (1.07E-07)	1.06E-07 (1.61E-07)	5.52E-08 (0.83E-09)	**7.67E-11 (5.48E-10)**
	SO-Tournament	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
	SO-Linearrank	7.10E-04 (0.12E-03)	7.10E-04 (0.88E-03)	4.58E-04 (0.48E+00)	0.11E-05 (5.71E-03)	**3.87E-07 (1.62E-07)**
	SO-Exponentialrank	2.27E+03 (8.36E+03)	2.08E+03 (1.00E+03)	0.35E+02 (5.00E+02)	0.02E+02 (1.52E+02)	**1.28E+01 (1.20E+00)**
Rosenbrock	SO-Global-best	0.02423 (0.8008)	0.11E-05 (0.27E-05)	1.44E-05 (6.58E-06)	5.53E-06 (3.22E-06)	**0.54E-07 (1.26E-08)**
	SO-Roulettewheel	0.20E-07 (1.15E-07)	7.20E-06 (1.15E-06)	5.47E-06 (3.14E-07)	**1.24E-08 (3.56E-07)**	2.07E-08 (5.18E-07)
	SO-Tournament	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
	SO-Linearrank	0.5586 (1.1032)	2.30E-02 (0.48E+01)	2.74E-04 (4.10E-04)	3.21E-05 (7.83E-06)	**1.71E-08 (6.54E-07)**
	SO-Exponentialrank	0.35E+04 (5.60E+03)	2.17E+03 (5.00E+03)	0.41E+03 (1.00E+03)	1.06E+02 (0.38E+03)	**0.05E+02 (1.00E+03)**
Rotated hyper-ellipsoid	SO-Global-best	4.44E-06 (8.05E-06)	8.10E-06 (2.13E-06)	4.44E-06 (8.05E-06)	4.58E-06 (1.33E-06)	**4.48E-07 (1.28E-07)**
	SO-Roulettewheel	6.50E-07 (0.12E-06)	6.50E-07 (0.12E-06)	2.540E-07 (6.21E-08)	5.73E-08 (6.45E-07)	**4.58E-09 (2.38E-10)**
	SO-Tournament	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
	SO-Linearrank	0.05E-04 (0.0078)	1.28E-04 (0.08E+01)	0.05E-06 (2.30E-06)	0.45E-05 (4.10E-06)	**2.74E-08 (4.30E-06)**
	SO-Exponentialrank	0.70E+06 (5.45E+05)	0.38E+03 (2.18E+06)	2.13E+03 (1.00E+02)	1.22E+02 (1.72E-02)	**5.83E+00 (0.05E+02)**
Schwefel's problem 2.26	SO-Global-best	-559.687870 (1.084706)	**-23675.928 (1.136271)**	-23560.799 (1.837641)	-23668.768 (1.023145)	-23661.314 (1.156238)
	SO-Roulettewheel	-955.3679 (01.8273)	-3628.645 (545.183740)	-23669.613 (8.422817)	-23671.654 (1.323535)	**-23674.796 (0.840532)**
	SO-Tournament	-941.927358 (146.046462)	**-23678.54 (2.03102)**	-23641.597 (0.000016)	-23641.512 (0.41E-04)	-23674.989 (0.07E-04)
	SO-Linearrank	-9865.935499 (288.744633)	-23672.53 (0.752077)	-23675.928 (1.136271)	-23664.454 (1.743721)	**-23677.965 (0.087465)**
	SO-Exponentialrank	-593.4963 (662.4268)	-9876.7576 (300.067265)	-9876.757 (300.067)	-9976.757 (130.421)	**-22894.654 (112.384)**
Rastrigin	SO-Global-best	8.58E-04 (0.61E-04)	5.83E-04 (8.80E-05)	5.10E-06 (2.46E-05)	2.25E-07 (5.23E-06)	**0.48E-07 (2.83E-07)**
	SO-Roulettewheel	5.76E-07 (0.20E-07)	5.74E-08 (0.20E-07)	7.06E-08 (6.61E-08)	1.43E-08 (0.77E-08)	**3.47E-09 (3.48E-08)**
	SO-Tournament	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
	SO-Linearrank	5.58E-03 (3.72E-03)	0.28E-03 (0.28E-03)	4.58E-04 (0.48E-03)	2.00E-04 (7.11E-03)	**3.34E-05 (1.04E-04)**
	SO-Exponentialrank	0.77E+05 (1.54E+05)	0.83E+05 (3.08E+05)	7.35E+04 (5.00E+03)	1.083E+02 (2.32E+04)	**6.58E+02 (1.04E+02)**

Table 3: The Average and Standard Deviation Results of Benchmark Functions.

Benchmark Function	Selection Technique	Scenario1	Scenario2	Scenario3	Scenario4	Scenario5
Ackley	SO-Global-best	4.54E-03 (3.15E-03)	1.35E-03 (4.23E-03)	0.01E-03 (5.43E-04)	6.55E-04 (1.27E-05)	**5.34E-05 (8.21E-05)**
	SO-Roulettewheel	8.26E-04 (1.60E-03)	4.47E-05 (6.74E-05)	0.03E-05 (1.64E-05)	7.01E-05 (2.44E-06)	**2.32E-07 (7.45E-08)**
	SO-Tournament	3.77E-04 (0.00E+00)	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00
	SO-Linearrank	6.1207 (4.1086)	3.4124 (6.56E-02)	2.3824 (1.15E-03)	0.5586 (0.00E-03)	**0.1738 (5.55E-3)**
	SO-Exponentialrank	18.5480 (1.7650)	17.1142 (0.681)	17.1344 (6.18E-02)	16.3562 (2.21E-04)	**12.8726 (0.16E-04)**
Griewank	SO-Global-best	5.24E-05 (0.27E-06)	1.27E-05 (4.78E-05)	0.54E-06 (6.07E-06)	**2.12E-08 (7.52E-07)**	3.43E-08 (6.24E-07)
	SO-Roulettewheel	1.27E-06 (4.78E-06)	4.70E-07 (6.33E-05)	4.70E-07 (6.33E-05)	5.74E-08 (7.20E-07)	**6.15E-09 (4.57E-09)**
	SO-Tournament	**0.00E+00 (0.00E+00)**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)
	SO-Linearrank	0.0452 (0.0721)	5.05E-03 (5.08E-04)	6.02E-05 (0.75E-04)	0.14E-06 (6.32E-05)	**0.01E-07 (1.34E-05)**
	SO-Exponentialrank	0.23E+03 (0.65E+03)	3.85E+02 (5.54E+02)	2.76E+02 (1.86E+02)	2.03E+02 (1.86E+02)	**1.00E+00 (0.01E+00)**
Camel-Back	SO-Global-best	-0.9386 (0.086)	-8.99E-01 (2.13E-01)	**-9.18E-01 (3.73E-02)**	-9.01E-01 (2.83E-02)	-8.78E-01 (2.15E-01)
	SO-Roulettewheel	-0.8784 (0.1064)	-8.99E-01 (4.010E-01)	-9.55E-01 (8.763E-01)	**-9.85E-01 (8.76E-02)**	-9.20E-01 (1.01E-02)
	SO-Tournament	-0.8467 (0.00E+00)	-6.45E-01 (2.01E-01)	-9.89E-01 (6.03E-04)	-9.89E-01 (6.03E-04)	**-9.87E-01 (9.76E-04)**
	SO-Linearrank	0.6278 (1.1030)	0.42E+00 (2.52E+00)	0.01E+00 (0.02E+03)	0.01E+00 (0.03E+03)	**-9.99E-01 (0.65E-03)**
	SO-Exponentialrank	6.11E+03 (7.17E+03)	8.23E+03 (0.04E+04)	4.75E+03 (8.63E+02)	4.75E+03 (8.63E+02)	**2.76E+02 (8.63E-02)**
Shifted Sphere	SO-Global-best	0.63E+04 (5.10E+03)	4701.5781 (1650.0532)	176.1707 (1074.8632)	-440.967 (1.5315)	**-448.735 (0.2537)**
	SO-Roulettewheel	0.74E+04 (5.00E+03)	-445.9375 (0.0171)	-447.5478 (0.6462)	-442.8980 (0.3451)	**-449.0970 (1.3456)**
	SO-Tournament	1.53E+04 (0.02E+04)	**-449.9887 (0.0076)**	-447.8877 (0.0008)	-448.6540 (0.3256)	-449.9690 (0.0652)
	SO-Linearrank	2.16E+05 (7.83E+04)	4701.5681 (1650.0532)	2454.5217 (1065.5807)	642.4270 (6.22E+05)	**631.6450 (1.28E+05)**
	SO-Exponentialrank	0.83E+06 (3.80E+06)	4.70E+05 (2.65E+06)	1.46E+05 (1.07E+06)	8.45E+04 (5.47E+05)	**5.21E+04 (5.54E+04)**
Shifted Schwefel's problem 1.2	SO-Global-best	4.30E+04 (0.15E+04)	835487.6950 (208027.653)	-439.9446 (0.041105)	-440.9780 (0.8532)	**-449.772 (0.06543)**
	SO-Roulettewheel	4.03E+04 (0.28E+04)	-48.8595 (255.0173)	-449.7793 (255.01733)	-441.799 (0.21E-02)	**-449.9310 (1.58E-03)**
	SO-Tournament	6.87E+04 (2.24E+04)	-447.0061 (1.8581)	-449.8697 (5.3633)	-448.473 (0.27E-03)	**-449.959 (6.08E-03)**
	SO-Linearrank	0.76E+06 (8.07E+05)	-449.9446 (0.04110)	-216.7574 (0.0411)	-421.4110 (7.30E-03)	**-439.654 (5.54E-03)**
	SO-Exponentialrank	4.60E+04 (0.40E+04)	5487.5840 (8927.6540)	1532.7480 (6475.0145)	3074.0130 (2767.3530)	**435.0010 (447.2040)**
Shifted Rosenbrock	SO-Global-best	1.01E+12 (1.13E+11)	404.0800 (104.5541)	408.4310 (252.2410)	**400.2102 (104.5421)**	401.0367 (201.4681)
	SO-Roulettewheel	1.03E+12 (1.50E+11)	386.0000 (105.7600)	487.6550 (115513)	468.004 (1.75E+03)	**454.6333 (1.00E+03)**
	SO-Tournament	1.38E+12 (5.43E+11)	475.7140 (110.4870)	490.4560 (122.8890)	420.1020 (321.5670)	**378.5490 (202.321)**
	SO-Linearrank	1.16E+13 (5.31E+12)	478.3000 (308.8300)	595.4780 (108.3210)	570.1234 (281.7543)	**543.4710 (218.5000)**
	SO-Exponentialrank	1.12E+12 (1.50E+11)	1405.7900 (1.76E+8)	1100.5431 (1.45E+7)	980.6789 (1.61E+05)	**870.3210 (1.60E+05)**
Shifted Rastrigin	SO-Global-best	1322.0560 (25.7743)	**-329.9980 (0.2045)**	-329.1239 (0.9590)	-302.9870 (0.2169)	-319.8900 (0.4321)
	SO-Roulettewheel	133.0134 (20.6035)	-329.9620 (0.184239)	**-429.8760 (0.1742)**	-320.8876 (1.4310)	-323.7890 (0.2310)
	SO-Tournament	132.5543 (16.8017)	-220.0990 (12.0074)	-328.7689 (1.7653)	-329.7890 (3.50E-03)	**-429.8890 (2.61E-03)**
	SO-Linearrank	7.68E+03 (170.0245)	-329.967 (0.0731)	**-389.0990 (1.6091)**	-289.8860 (0.2361)	-320.9780 (0.6538)
	SO-Exponentialrank	922.6654 (46.2210)	1025.7654 (1765.2100)	1743.3210 (1542.9900)	-201.6541 (38.5210)	**-281.5678 (3.7651)**

REFERENCES

Abu Khurma, R., Aljarah, I., Sharieh, A., Abd Elaziz, M., Damaševičius, R., and Krilavičius, T. (2022). A review of the modification strategies of the nature inspired algorithms for feature selection problem. *Mathematics*, 10(3):464.

Adam, S. P., Alexandropoulos, S.-A. N., Pardalos, P. M., and Vrahatis, M. N. (2019). No free lunch theorem: A review. *Approximation and optimization*, pages 57–82.

Al-Betar, M. A., Awadallah, M. A., Faris, H., Aljarah, I., and Hammouri, A. I. (2018a). Natural selection methods for grey wolf optimizer. *Expert Systems with Applications*, 113:481–498.

Al-Betar, M. A., Awadallah, M. A., Faris, H., Yang, X.-S., Khader, A. T., and Alomari, O. A. (2018b). Bat-inspired algorithms with natural selection mechanisms for global optimization. *Neurocomputing*, 273:448–465.

Al-Betar, M. A., Doush, I. A., Khader, A. T., and Awadallah, M. A. (2012). Novel selection schemes for harmony search. *Applied Mathematics and Computation*, 218(10):6095–6117.

Awadallah, M. A., Al-Betar, M. A., Bolaji, A. L., Alsukhni, E. M., and Al-Zoubi, H. (2019). Natural selection methods for artificial bee colony with new versions of onlooker bee. *Soft Computing*, 23(15):6455–6494.

Civicioglu, P. and Besdok, E. (2013). A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. *Artificial intelligence review*, 39(4):315–346.

Hashim, F. A. and Hussien, A. G. (2022). Snake optimizer: A novel meta-heuristic optimization algorithm. *Knowledge-Based Systems*, 242:108320.

Jin, X., Liang, Y., Tian, D., and Zhuang, F. (2013). Particle swarm optimization using dimension selection methods. *Applied Mathematics and Computation*, 219(10):5185–5197.

Khurma, R. A., Aljarah, I., and Sharieh, A. (2021). A simultaneous moth flame optimizer feature selection approach based on levy flight and selection operators for medical diagnosis. *Arabian Journal for Science and Engineering*, 46(9):8415–8440.

Khurma, R. A., Aljarah, I., Sharieh, A., and Mirjalili, S. (2020). Evolopy-fs: An open-source nature-inspired optimization framework in python for feature selection. In *Evolutionary machine learning techniques*, pages 131–173. Springer.

Li-Ping, Z., Huan-Jun, Y., and Shang-Xu, H. (2005). Optimal choice of parameters for particle swarm optimization. *Journal of Zhejiang University-Science A*, 6(6):528–534.

Richards, M. and Ventura, D. A. (2003). Dynamic sociometry in particle swarm optimization.

Shehab, M., Khader, A. T., and Al-Betar, M. (2016). New selection schemes for particle swarm optimization. *IEEJ Transactions on Electronics, Information and Systems*, 136(12):1706–1711.

Trelea, I. C. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, 85(6):317–325.