A Modified Spider Monkey Optimization Algorithm Based on Good-Point Set and Enhancing Position Update

Sabreen Fawzi Raheem[1], Maytham Alabbas[2*]
[1] Basra Technical Institute, Southern Technical University, Basrah, Iraq
[2] Department of Computer Science, College of Computer Science and Information Technology, University of Basrah, Basrah, Iraq
E-mail: sabreen.fawzi@stu.edu.iq[1], ma@uobasrah.edu.iq[2*]
[*] Corresponding author

Keywords: good-point-set method, swarm intelligence, enhancing position update, spider monkey optimization

Received: November 23, 2022

Spider monkey optimization (SMO), developed based on the behavior of spider monkeys, is a recently added swarm intelligence algorithm. SMO is a stochastic population-based metaheuristic algorithm. Spider monkeys have a fission-fusion social structure. In addition to being an excellent tool for solving optimization problems, SMO provides good exploration and exploitation capabilities. In this work, we present a modified strategy for improving the performance of the basic SMO in two ways: (a) we use the good-point-set method instead of random initial population generation; and (b) by changing both the local leader and global leader phases, we modify the SMO position update approach to increase global convergence while avoiding local optima. We evaluate the proposed modified SMO algorithm on 20 popular benchmark functions. The current findings prove that the proposed approach outperforms the standard SMO regarding result quality and convergence rate.

Povzetek: Predstavljeni SMO, stohastičen algoritem, ki temelji na socialni strukturi pajkovih opic, izboljšuje osnovno različico z dvema pristopoma: metodo dobre točke in modificiranim posodabljanjem položaja za večjo globalno konvergenco.

1 Introduction

Nature has inspired many researchers and is therefore considered a rich source of inspiration. Nowadays, most algorithms are inspired by nature. These algorithms depend primarily on one of the successful properties of a biological system. Numerous natural phenomena have inspired academics to create population-based optimization (PBO) methods. Because these PBO methods evaluate fitness, the population of possible solutions is expected to gravitate toward the areas of the potential search spaces with the best fitness. In PBO algorithms, natural selection offers near-optimal solutions to complex optimization problems.

In recent years, researchers have gained interest in swarm intelligence (SI). Studies have proven that algorithms based on SI have enormous potential for numerical optimization. The last few years have seen the development of many related algorithms. In this regard, there are several algorithms available, but not limited to particle swarm optimization (PSO) [1], ant colony optimization (ACO) [2], cuckoo search [3], bacteria foraging optimization (BFO) [4], bat algorithm [5], and artificial bee colony (ABC) [6].

Among the latest techniques to be developed is spider monkey optimization (SMO) [7], a newcomer to the class of SI algorithms. Spider monkey foraging behavior based on fission-fusion social structure (FFSS), which they use to find great food sources and mate, served as the model for this SMO algorithm. In the same way as any other PBO strategy, it includes the intrinsic population solution that denotes the spider monkey food source.

While searching for the best solution, the SMO algorithm tries to find a suitable balance between exploration and exploitation. It ensures that the local optimal solution is traversed correctly during exploitation, and it explores the global search space to prevent the problem of entrapment in the local optimum during exploration. It has been discovered that SMO effectively explores local search [8].

As SMO is a relatively new algorithm, there is little literature available on it. The authors in [9] developed a modified position update in the SMO method, which includes two revisions to the basic SMO algorithm. The golden section search (GSS) technique was used to modify both the local leader phase (LLP) and the global leader phase (GLP). In [10], Kumar et al. proposed individual fitness as a new technique for updating the position of spider monkeys in the LLP, GLP, and local leader decision (LLD) phases. In [11], Kavita and Kusum proposed a modified version of SMO (TS-SMO), which uses tournament selection to increase the exploration capabilities of SMO and prevent premature convergence. Urvinder and Rohit [12] introduced a modified SMO (MSMO) method based on a dual-search technique for linear antenna array synthesis (LAA). Singh et al. [13] suggested a modified version of SMO (SMONM), which uses Nelder-Mead (NM) transformations to improve the ability of the LLP. The proposed SMONM approach has

the same number of phases as the conventional SMO, except for the LLP, which is modified using the NM reflection, expansion, and contraction transformations. If there is no improvement in the fitness function value after updating the solution with the original LLP, NM modifications are applied.

The rest of this paper is organized as follows: Section 2 briefly explains the fundamental SMO algorithm. Section 3 describes the primary concept of the proposed approach. Section 4 discusses the test functions, simulation results, and comparison outcomes. Section 5 provides a comprehensive explanation.

2 Spider monkey optimization (SMO)

Bansal et al. [7] introduced the SMO algorithm, a nature-inspired evolutionary algorithm. The fission-fusion social structure (FFSS) of spider monkeys (SMs) is modeled in SMO. SMs live in groups of forty to fifty monkeys and separate into subgroups to look for food to reduce competition. The group is led by the most senior female, who is responsible for finding food sources. If she cannot find enough food for the group, she divides it into small subgroups of three to eight individuals. The local leader (LL) leads the subgroups and plans their foraging paths each day. The members of this group are responsible for finding food sources and adjusting their location according to the distance to the food source. It is important for these group members to interact with each other to maintain social bonds, especially if they encounter a stalemate.

2.1 The SMO algorithm's phases

SMO consists of seven phases. Below is a detailed description of each phase of SMO. The steps of the SMO algorithm's phases are given in Algorithm 1.

2.1.1 Initialization phase

Initially, SMO produces N of SMs with uniform distribution. Each SM_i = 1, .., N is a D-dimensional vector. The number D indicates how many variables are involved in the optimization problem, and SM_i is the i^{th} individual in the population. Each individual is generated using Eq. 1.

$$SM_i = Lb_j + rand(0,1) * \left(Ub_j - Lb_j\right), \forall j \in \{1, \dots, D\}, \forall i = \{1, \dots, N\}, \quad (1)$$

where *rand(0,1)* is a random number between *0* and *1*, the lower bound of the solution location is given by Lb_j, and the upper bound of the solution location is given by Ub_j.

2.1.2 Local leader phase (LLP)

During LLP, SMs change their present positions depending on information from the LL's experience as well as the experience of local group members. The fitness value of the newly acquired location is computed. If the new position's fitness value exceeds the old position's, the SM replaces its position with the new one. In this phase, the position update equation for the i^{th} SM (member of the k^{th} local group) is calculated as follows:

$$SMnew_{ij} = SM_{ij} + rand(0,1) * \left(LL_{kj} - SM_{ij}\right) + rand(-1,1) * \left(SM_{rj} - SM_{ij}\right), \quad (2)$$

where LL_{kj} stands for the j^{th} dimension of the k^{th} local group leader position, SM_{ij} stands for the j^{th} dimension of the i^{th} SM, and SM_{rj} is the j^{th} dimension of the k^{th} SM that is selected at random within the k^{th} group, such that (r≠i).

2.1.3 Global leader phase (GLP)

The GLP phase begins once the LLP is completed. During this phase, each SM updates its position using Eq. 3, considering the experience of the global leader (GL) and local group members.

$$SMnew_{ij} = SM_{ij} + rand(0,1) * \left(GL_j - SM_{ij}\right) + rand(-1,1) * \left(SM_{rj} - SM_{ij}\right), \quad (3)$$

where GL_j is the j^{th} dimension of the GL position and $j \in \{1,2, \dots, D\}$ is a randomly selected index.

SM_i's location is updated throughout this phase based on the probability $prob_i$ that can be calculated using Eq. 4.

$$Prob_i = \frac{fitness_i}{\sum_{i=1}^{N} fitness_i}. \quad (4)$$

2.1.4 Global leader learning (GLL)

A greedy selection is used to update the GL's position. Consequently, the GL's position is updated based on the SM's position that has the best fitness. In addition, the *GlobalLimitCount (GLLimit)* is increased by one if the GL's position has not been updated.

2.1.5 Local leader learning (LLL)

A greedy selection method is used to update the position of the LL in that group, i.e., to reflect the position of the new LL. Selection is made based on the position of the SM in the group with the best fitness. A comparison is then conducted between the new position of the LL and the earlier one, and *LocalLimitCount (LLLimit)* is increased by one if no changes are made.

2.1.6 Local leader decision (LLD)

As long as no LL position has changed above a *LocalLeaderLimit* threshold, whether it is random or using Eq. 5, all group members are updated. This is determined by the perturbation rate (pr), which specifies how much perturbation is in the current position.

If *rand(0,1) >= pr* **then**
Using Eq. 1.
Else

$$SMnew_{ij} = SM_{ij} + rand(0,1) * (GL_j - SM_{ij}) + rand(0,1) * (SM_{ij} - LL_{kj}). \quad (5)$$

2.1.7 Global leader decision (GLD)

After a certain number of iterations, the global leader's (GL's) position is observed. If it is not changed, the GL divides the population into subgroups. Initially, two groups are separated from the population, and the number of groups is increased until it reaches the maximum number of groups (MG).

The pseudocode for the SMO algorithm is presented in Algorithm 1 [7].

Algorithm 1: SMO algorithm

Initialize the population, the Local Leader Limit (LLlimit), the Global Leader Limit (GLlimit), and the perturbation rate (pr).
Calculate fitness
Choose leaders (both global and local) through greedy selection.
While *(termination criteria is not)* **do**
Create new locations for all group members based on your own experience, local leader experience, and group member experience. Using Eq. (2)
Use the greedy selection process to choose between an existing location and a newly generated location based on fitness, and then choose the better one.
Using Eq. (4), compute the probability pi for each group member.
Create new locations for all group members based on pi's selection, using self-experience, global leader experience, and group member experiences. Using Eq. (3).
By applying the greedy selection process to all groups, you can update the position of local and global leaders.
If a Local group leader does not update her position after a set number of times (LLLimit), the algorithm will re-direct all members of that group to foraging.
If the Global Leader does not update her position for a set number of times (GLLimit), she divides the group into smaller groups using the steps below.
End while

3 The current work

In the current work, two modifications have been made to make the basic SMO algorithm work better. The first modification is that we used the good-point-set method to generate a suitable initial population for the SMO algorithm. Then, we modified both LLP and GLP of the basic SMO algorithm.

3.1 Modification of the initial population

Initializing the population is one of the most crucial steps in metaheuristic optimization. A successful initial population can generate reasonable solutions faster and accelerate convergence. The basic SMO algorithm randomly generates the initial population using the stochastic method. However, the initial solutions do not cover the entire search space in quality and may be good or bad. To address these challenges, we used the good-point-set (GPS) method [14] to generate the initial population of the SMO algorithm. GPS is generally used to create several individuals with uniform distributions to preserve the population's diversity. The GPS pseudocode is presented in Algorithm 2 [15, 16,17,18].

Algorithm 2: Pseudocode of the GPS method

*P is the lower prime number with P ≥ 2*D +3.*

For *i=0 to N* **do**
  For *j=0 to D* **do**
    $x = 2 * n + 3,$
    While *P ≠ x* **do**
      Set individual counter k = 2
      For *k = 2 to x – 1* **do**
        If *mod(x , i) == 0* **then**
          x = x + 1
        Else
          P = x
        End if
      End for
    End while
    $GPS_{i,j} = \mathrm{i} * 2 * \cos\left(2 * \pi * \frac{j}{P}\right)$
    $X_{ij} = Lb_j + GPS_{i,j} * (Ub_j - Lb_j)$
  End for
End for

3.2 Modification of the position update in SMO

Maintaining diversity in the LLP and GLP of SMO can balance exploring the entire search space and exploiting the best solutions found nearby. To achieve this balance, the proposed algorithm modifies both LLP and GLP of the basic SMO algorithm using the modified Eqs. 6 and 7, respectively.

$$SMnew_{ij} = SM_{ij} + rand(0,1) * (LL_{kj} - SM_{ij}) + rand(-1,1) * (SM_{r1j} - SM_{r2j}). \quad (6)$$

$$SMnew_{ij} = SM_{ij} + rand(0,1) * (GL_{kj} - SM_{ij}) + rand(-1,1) * (SM_{r1j} - SM_{r2j}). \quad (7)$$

Where $r1 \in \{1,2,\dots,N\}$ is a different index from i that was chosen at random from the population, and $r2 \in \{1,2,\dots,N\}$ is a different index from i, where $r1 \neq r2$.

4 Experiments

4.1 Test functions

The improved SMO algorithm is evaluated using 20 well-known benchmark optimization functions, f_1-f_{20}, as shown in Table 1. These continuous optimization problems cover a range of difficulties, search spaces, and multimodality.

Table 1: The current 20 benchmark optimization functions for testing

NO	Function	type	Search Range	optimal value	Formulation
f_1	Rosenbrock	UN	[-50,50]	0	$f_1(x) = \sum_{i=1}^{D-1} 100\,(x_{i+1} - x_i{}^2)^2 + (x_i - 1)^2$
f_2	sphere	US	[-100,100]	0	$f_2(x) = \sum_{i=1}^{D} x_i^2$
f_3	Elliptic	UN	[-100,100]	0	$f_3(x) = \sum_{i=1}^{D} (10^6)^{(i-1)/(D-1)} x_i^2$
f_4	Sum Squares	US	[-10,10]	0	$f_4(x) = \sum_{i=1}^{D} i\, x_i^2$
f_5	Quartic	US	[-1.28,1.28]	0	$f_5(x) = \sum_{i=1}^{D} i\, x_i^4$
f_6	kowalik	MS	[-5,5]	0.0003075	$f_6(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4} \right]^2$
f_7	Scaffer's F6	MN	[-100,100]	0	$f_7(x) = 0.5 + \frac{sin^2\left(\sqrt{\sum_{i=1}^{D} x_i^2}\right) - 0.5}{(1 + 0.001\,(\sqrt{\sum_{i=1}^{D} x_i^2}))^2}$
f_8	Rastrigin	MS	[-5.12,5.12]	0	$f_8(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$
f_9	Griewank	MN	[-600,600]	0	$f_9(x) = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
f_{10}	Ackley	MN	[-32.768,32.768]	0	$f_{10}(x) = 20 + e - 20exp\left[-0.2 \sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right] - exp\left[\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi\, x_i)\right]$

f_{11}	Himmelblau	MS	[-5,5]	78.3323	$f_{11}(x) = \frac{1}{D}\sum_{i=1}^{D}(x_i^4 - 16\,x_i^2 + 5\,x_i$
f_{12}	Step	US	[-100,100]	0	$f_{12}(x) = \sum_{i=1}^{D}(\lfloor x_i + 0.5 \rfloor_i)^2$
f_{13}	Beale	UN	[- 4.5,4.5]	0	$f_{13}(x) = [1.5 - (1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2$
f_{14}	Easom	UN	[-100,100]	-1	$f_{14}(x) = -\cos x_1 \cos x_2 \; e^{((-(x_1-\pi)^2(-(x_2-\pi)^2)}$
f_{15}	Schwefel	MS	[-500,500]	0	$f_{15}(x) = 418.9829 * D - \sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$
f_{16}	Levy N.13	MN	[-10,10]	0	$f_{16}(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2(1 + \sin^2(3\pi x_2)) + (x_2 - 1)^2(1 + \sin^2(2\pi x_2))$

f_{17}	Goldstein-Price	MN	[-2,2]	3	$f_{17}(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 13x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] * [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 - 48x_2 - 36x_1x_2 + 27x_2^2)$
f_{18}	Colville	MN	[-10,10]	0	$f_{18}(x) = [100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1((x_2 - 1)^2 + (x_4 - 1)^2) + 19.8(x_2 - 1)(x_4 - 1)]$
f_{19}	Booth	UN	[-10,10]	2	$f_{19}(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
f_{20}	Dekkers and Aarts	MN	[-20,20]	2	$f_{20}(x) = 10^5 + x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-5}(x_1^2 + x_2^2)^4$

4.2 Parameter setting

The present work and the basic SMO share the following primary control parameters:

- The Swarm size (N) = 80,
- The Minimum Group (MG) = 5,
- LocalLeaderLimit = N
- GlobalLeaderLimit = D * N,
- Pr grows linearly over iterations by Eq. 8, where initial pr ∈ [0.1, 0.4],

$$pr = pr + (0.4/iter). \quad (8)$$

4.3 Experimental results

To evaluate the proposed approach, we compared it to the basic SMO algorithm on 20 popular benchmark functions. Table 2 shows the mean and standard deviation (SD) of the solutions obtained by each algorithm based on 15 runs. The best outcomes highlighted in bold typeface.

As shown in Table 2, the proposed approach outperforms the basic SMO algorithm in most cases in terms of convergence rate, convergence speed, and global optimization capability.

5 Conclusions

This work proposes two modifications to address the shortcomings of the basic SMO algorithm. The first modification improves the distribution of the initial population using the GPS method during the initialization phase. The second modification modifies the LLP and GLP phases to account for the intensified and diversified breathing space for local searches. We evaluated the proposed algorithm on 20 standard benchmark functions. The simulation results show that the proposed algorithm outperforms the basic SMO algorithm and provides accurate and robust solutions.

We aim to improve the performance of the modified SMO algorithm in a number of ways using powerful soft computing tools. By using soft computing tools, we can control the parameters of the SMO algorithm, guide the

search process, and develop hybrid algorithms to create algorithms that are more efficient and effective than the basic SMO algorithm.

Table 2: Comparison of test problem results

Function	D	Basic SMO		Modified SMO	
		Mean	SD	Mean	SD
Rosenbrock	10	967.95828	2311.63421	**7.93559**	0.34621
	30	28.66994	1.93780	**27.33593**	0.36470
sphere	30	2.24285e-13	8.39023e-13	**2.89926e-17**	8.26299e-18
	60	7.85583e-17	7.66269e-18	**2.63342e-17**	7.62556e-18
Elliptic	30	5.75254e-17	9.29459e-17	**2.78168e-17**	7.55754e-18
	60	7.63723e-17	7.43771e-18	**3.06328e-17**	4.73009e-18
Sum Squares	30	7.62003e-10	2.85115e-09	**3.04396e-17**	7.97876e-18
	60	7.02444e-17	1.20772e-17	**3.74735e-17**	6.21793e-18
Quartic	30	**1.16326e-23**	4.03573e-23	1.49493e-23	4.31037e-23
	60	**2.04903e-23**	4.45292e-23	1.15639e-22	1.99527e-22
Kowalik	4	0.00105094	0.0006879576	**0.00032137**	6.26241e-05
Schaffer's F6	10	0.03602	0.01491	**0.00906**	0.00242
	30	0.03327	0.00898	**0.00972**	1.03182e-09
Rastrigin	30	114.81071	19.08043	**0**	0
	60	103.08621	67.12331	**0**	0
Griewank	30	1.39276e-06	4.74962e-06	**0**	0
	60	9.75133e-11	2.09656e-10	**0**	0
Ackley	30	9.37203e-12	3.50519e-11	**3.28626e-15**	1.42108e-15
	60	4.23365e-15	8.86203e-16	**3.99681e-15**	0
Himmelblau	100	**- 78.3323**	9.07×10^{-12}	**- 78.3323**	3.48×10^{-14}
Step	30	**0**	**0**	**0**	**0**
Beale	2	$\mathbf{1.33 \times 10^{-13}}$	4.00×10^{-16}	4.91×10^{-8}	2.36×10^{-8}
Easom	2	**-1**	0	**-1**	9.05×10^{-13}
Schwefel	30	$\mathbf{3.82 \times 10^{-4}}$	7.28×10^{-13}	$\mathbf{3.82 \times 10^{-4}}$	6.78×10^{-13}
Levy N.13	2	9.03×10^{-20}	7.79×10^{-20}	$\mathbf{1.97 \times 10^{-20}}$	1.72×10^{-20}
Goldstein-Price	2	**3**	4.81×10^{-8}	**3**	1.1×10^{-15}
Colville	4	5.18×10^{-2}	5.59×10^{-2}	$\mathbf{8.24 \times 10^{-3}}$	1.83×10^{-2}
Booth	2	1.75×10^{-18}	1.92×10^{-12}	1.33×10^{-18}	1.15×10^{-18}
Dekkers and Aarts	2	**-24776.52**	7.27×10^{-12}	**-24776.52**	7.28×10^{-12}

References

[1] M. Clerc, Particle swarm optimization. John Wiley & Sons, 2010.

[2] M. Dorigo and T. Stützle, "Ant colony optimization: overview and recent advances," Handbook of metaheuristics, pp. 311-351, 2019.

[3] M. Shehab, A. T. Khader, and M. A. Al-Betar, "A survey on applications and variants of the cuckoo search algorithm," Applied Soft Computing, vol. 61, pp. 1041-1059, 2017. https://doi.org/10.1016/j.asoc.2017.02.034

[4] C. Guo, H. Tang, B. Niu, and C. B. P. Lee, "A survey of bacterial foraging optimization," Neurocomputing, vol. 452, pp. 728-746, 2021. https://doi.org/10.1016/j.neucom.2020.06.142

[5] X.-S. Yang and X. He, "Bat algorithm: literature review and applications," International Journal of Bio-inspired computation, vol. 5, no. 3, pp. 141-149, 2013. https://doi.org/10.1504/IJBIC.2013.055093

[6] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," TECHNICAL REPORT-TR06 2005.

[7] J. C. Bansal, H. Sharma, S. S. Jadon, and M. Clerc, "Spider monkey optimization algorithm for numerical optimization," Memetic computing, vol. 6, no. 1, pp. 31-47, 2014.

[8] K. Gupta, K. Deep, and J. C. J. C. I. Bansal, "Improving the local search ability of spider monkey optimization algorithm using quadratic approximation for unconstrained optimization," vol. 33, no. 2, pp. 210-240, 2017. https://doi.org/10.1111/coin.12081

[9] S. Kumar and R. Kumari, "Modified position update in spider monkey optimization algorithm," in International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS, 2014: Citeseer.

[10] S. Kumar, R. Kumari, and V. K. Sharma, "Fitness based position update in spider monkey optimization algorithm," Procedia Computer Science, vol. 62, pp. 442-449, 2015. https://doi.org/10.1016/j.procs.2015.08.504

[11] K. Gupta and K. Deep, "Tournament selection based probability scheme in spider monkey optimization algorithm," in Harmony Search Algorithm: Springer, 2016, pp. 239-250. DOI: 10.1007/978-3-662-47926-1_23

[12] U. Singh and R. Salgotra, "Optimal synthesis of linear antenna arrays using modified spider monkey optimization," Arabian Journal for Science and Engineering, vol. 41, no. 8, pp. 2957-2973, 2016.

[13] P. R. Singh, M. Abd Elaziz, and S. Xiong, "Modified Spider Monkey Optimization based on Nelder–Mead method for global optimization," Expert Systems with Applications, vol. 110, pp. 264-289, 2018. https://doi.org/10.1016/j.eswa.2018.05.040

[14] H. Liu, Z. Cai, and Y. Wang, "A new constrained optimization evolutionary algorithm by using good point set," in 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 1247-1254: IEEE. DOI: 10.1109/CEC.2007.4424613

[15] S. F. Raheem and M. Alabbas, "Dynamic Artificial Bee Colony Algorithm with Hybrid Initialization Method," Informatica, vol. 45, no. 6, 2021. https://doi.org/10.31449/inf.v45i6.3652

[16] M. Tang, W. Long, H. Wu, K. Zhang, and Y. A. W. Shardt, "Self-Adaptive Artificial Bee Colony for Function Optimization," Journal of Control Science and Engineering, vol. 2017, p. 13 pages, 2017. https://doi.org/10.1155/2017/4851493

[17] Athraa Qays Obaid, Maytham Alabbas, “Hybrid Variable-Length Spider Monkey Optimization with GoodPoint Set Initialization for Data Clustering”, Informatica 47 (2023) 67–78, https://doi.org/10.31449/inf.v47i8.4872.

[18] S. F. Raheem and M. Alabbas, "Optimal k-means clustering using artificial bee colony algorithm with variable food sources length," *International Journal of Electrical Computer Engineering*,vol. 12, no. 5, 2022. DOI: 10.11591/ijece.v12i5.pp5435-5443.