2021 43rd Annual International Conference of the
IEEE Engineering in Medicine & Biology Society (EMBC)
Oct 31 - Nov 4, 2021. Virtual Conference

A Novel Optimization Algorithm Leveraging a Three-Dimensional Approach of Periscopic, Pheromonic and Fractal Search

Rahul Dasharath Gavas, Venkatasubramanian Viraraghavan, Ramesh Kumar Ramakrishnan
TCS Research, India

***Abstract*— This paper focuses on a new algorithm for solving optimization problems using the nature of food search behaviour of caterpillars. The paper describes how the periscopic, pheromonic and fractal search properties analogous to the caterpillars, can aid in designing a new optimization algorithm. The performance characteristics of the new method is compared using 26 standard test functions and the area under the curve of the fitness evaluations is used to validate and compare the proposed algorithms against existing related works. The proposed algorithm is found to be efficient when compared with the existing methods. The proposed algorithm is then tested on a real world problem to remove signal noise from eye gaze data, effectively.**

I. Introduction

Nature has inspired computing solutions to several complex problems [1], [2], [3], [4], [5], [6], [7], [8]. Judging by the number of algorithms proposed, nature inspires solutions seems to work well for optimization problems. In general, an optimization problem involves a mathematical or algorithmic process of finding the best combination of parameters of a device or experiment that minimizes or maximizes its output, defined by a target-, cost-, or fitness-function [2]. However, an exhaustive search across possible parameter-values and ranges is impractical. Typical solutions reduce the search space, but can settle at local optima instead of the global optimum. Metaheursitc algorithms increase the probability of finding the global optimum solution. They do so by striking a balance between the spread of randomly searching various regions of the search space and refining the solutions in each region.

Meta-heuristics can draw upon physical laws (electromagnetic force, gravity etc.), evolutionary processes (reproduction, recombination, selection, and mutation), or swarm behaviour. Some optimization techniques of the last type are, Differential Evolution (DE) [3], Bat-inspired Algorithm (BA) [4], Grey Wolf Optimizer (GWO) [5], Whale Optimization Algorithm (WOA) [6], Crow Optimization [7], and Butterfly optimization [8]. In this paper, a new meta-heuristic algorithm inspired by the food search behaviour of caterpillar worms is proposed.

II. Methodology

Most of the caterpillars species are herbivorous (folivorous) in nature, but some (around 1%) are insectivorous and some are even cannibalistic. Some caterpillars feed on other animal products; for instance, cloth moths feed on wool, while horn moths feed on the hooves and horns of dead ungulates. They are typically voracious feeders. Most of them are serious agricultural pests.

The movement or the dispersal of caterpillars is governed by the characteristics of natural surfaces which has a huge impact on their reproductive and survival success [9], as well. Along with this, the pheromone of caterpillars guide their peers in the food search task. The nature of the surface is assessed using fractal analysis as the surface area and distance values are basically a function of the scale at which the measurements are made. The motion of caterpillars is dependent on the roughness of the surface and this particular aspect is incorporated in the proposed scheme using fractal dynamics of the solution space. Another important movement behaviour of caterpillars is the waving of their body. They lift their bodies upto a certain height and wave themselves in order to look for food in their vicinity. This nature along with the fractal analysis and the usage of pheromone is used to create the proposed algorithm.

A. Proposed Caterpillar Search Algorithm

The proposed method is given in Algorithm 1. The number of worms selected is denoted as N each with length L_w. The worms wave their bodies randomly in periscopic fashion to glance their neighbourhood and this probability is defined as P_w and follow their nearby worms using the pheromone. The roughness/fractality of the surface that determines the

Algorithm 1: Caterpillar Food Search Algorithm

```
Randomly initialize the positions of N worms in the search space
W = Initialize the state for each worm
while termination condition is not met do
    Compute α for fitness membership values (II-B)
    for i = 1 to N worms do
        Choose one worm randomly, denoted by s_i
        Define worm's waving probability r_i
        if r_i ≥ P_w then
            Evaluate H_w according to (3)
            u_i = w_i + rand() × H_w × (s_i − w_i × α)
        end
        else
            u_i = rand() × α
        end
    end
    Update W from U based on (4)
end
```

movement of the worms is given by the tuning parameter α computed using Kat'z algorithm [10].

1) Initialization of parameters: The positions of the N worms in D-dimensional search space are represented by W as given in 1.

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,d} \\ w_{2,1} & w_{2,2} & \dots & w_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ w_{N,1} & w_{N,2} & \dots & w_{N,d} \end{bmatrix} \quad (1)$$

Every worm $w_{i,*} \in W$ is a feasible solution and the number of decision variables (dimensions) is represented by D. For brevity, $w_{i,*}$ is denoted by w_i. Initially, the worms are placed at random positions. Next, the fitness value of each worm is computed by using its decision variable values in the objective function under test. This is carried out for every worm in W.

2) Update worm position: The worms update their positions in this phase to the potential state U. The parameter r_i is a random number in the range (0,1) and if this value is greater than P_w, then it is taken as an instance of the worm waving its body in a periscopic way. The worm then updates its path using the relation in equation 2. When a given worm i waves its body, it gets an idea of the nearby vicinity and follows the path of the worm j due to its pheromone (j is selected randomly). Its movement is determined by the waving height and the fractal value α (II-B) of the fitness functions traced by N worms as follows,

$$u_i = w_i + rand() \times H_w \times (w_j - w_i \times \alpha) \quad (2)$$

Initially, in the first run, the α value is taken as 0.5 as the fitness function evaluations are not present. Note that u_i are rows of U just as w_i are rows of W. The waving height H_w is different every time and we model it as follows,

$$H_w = L_w + (rand() + 1) \times 0.5 \quad (3)$$

with L_w being the length of the worm obtained empirically. The factor (rand() + 1)*0.5 ensures that the worm lifts at least 50% of this total length during waving phase. Hence, the mean waving height is 75% of the worm's total length. If the worm doesn't wave, i.e. $r_i \leq P_w$, then the position of that worm is reset randomly. After updating the worm positions, each of the new worms is tested and the fitness value $f(\cdot)$, is obtained. The fitness values are then subjected to fractal value computation and this value is used in the next iteration.

3) Update state of the worm: The state W is updated from a potential new state, U, as follows for each $i \in \{1, \dots, N\}$.

$$w_i = \begin{cases} u_i & \text{if } f(u_i) \text{ is better than } f(w_i) \\ w_i & \text{otherwise} \end{cases} \quad (4)$$

The above process is carried out until any one of the termination criteria is met. The criteria can be either of the following: (i) the number of function evaluations made, and (ii) if the best solution obtained is lower than a tolerance value for global minima, or higher than the tolerance value for global maxima.

TABLE I

DETAILS OF THE TEST FUNCTIONS USED FOR THE STUDY

Function id	Name	Lower bound	Upper bound	Dimensions
f1	Beale	-4.5	4.5	2
f2	Easom	-100	100	2
f3	Matyas	-10	10	1
f4	Bohachvesky 1	-100	100	2
f5	Booth	-10	10	2
f6	Michalewicz 2	0		2
f7	Schaffer	-100	100	2
f8	Six Hump Camel Back	-5	5	2
f9	Bohachvesky 2	-100	100	2
f10	Bohachvesky 3	-100	100	2
f11	Shubert	-10	10	2
f12	Colville	-10	10	4
f13	Michalewicz 5	0		5
f14	Zakharov	-5	10	10
f15	Michalewicz 10	0		10
f16	Step	-100	100	30
f17	Sphere	-100	100	30
f18	Sum Squares	-10	10	30
f19	Quartic	-1.28	1.28	30
f20	Schwefel 2.22	-10	10	30
f21	Schwefel 1.2	-100	100	30
f22	Rosenbrock	-30	30	30
f23	Dixon-Price	-10	10	30
f24	Rastringin	-5.12	5.12	30
f25	Griewank	-600	600	30
f26	Ackley	-32	32	30

B. Assessment of fractal properties

This section briefs the extraction of the fractal nature of the data using Katz Fractal Dynamics (FD) Algorithm [10]. The Katz FD value is given by,

$$\alpha = \frac{log_{10}(n)}{log_{10}(\frac{d}{L}) + log_{10}(n)} \quad (5)$$

where d is the euclidean distance between the first point in the time series and the point lying at the farthest distance; L is the total length of the data. Let a be the average distance between the successive points in the data then, the number of steps in the curve is given by, $n = \frac{L}{a}$. The fitness values computed for all worms are treated as a time series whose Katz FD value is evaluated.

C. Test functions used

We selected 26 standard test functions [3], [4], [5], [6], [7], [8] for evaluating our proposed optimization algorithm. The details of each of the functions with the parameters used is presented in Table I. These test functions cover a large spectrum of evaluations that can aid in benchmarking the designed optimization algorithm.

III. RESULTS AND DISCUSSIONS

We designed a metric to analyse the effectiveness of the optimization algorithms which is basically the measure of area under the curve (AUC) of the convergence curve for the given algorithm over the predefined number of iterations (bounded by the maximum value obtained in the first run and the solution obtained by that particular algorithm at the end of the predefined number of iterations). This can be visualized in Fig. 1. The AUC is the region bounded by

TABLE II

MEAN (SD) OF AUC VALUES ACROSS 30 RUNS. BEST PERFORMER IN BOLD. PROPOSED METHOD IS THE BEST FOR 15/26 FUNCTIONS.

F id	Proposed	Crow [7]	Bat [4]	Whale [6]	Butterfly [8]	Grey Wolf [5]	DE [3]
f1	12.99 (14.51)	1.4E+9 (6.07E+9)	1773.74 (3662.86)	167.31 (389.57)	717.92 (588.68)	58.95 (277.44)	**8.07 (7.22)**
f2	25.3 (8.3)	31.3 (5.6)	1828 (520.2)	30.7 (29.1)	1811.5 (721.1)	**9.75 (2.3)**	57.6 (21)
f3	**0.43 (0.29)**	16.23 (29.08)	189.95 (200.56)	0.78 (0.71)	14.72 (12.22)	0.83 (0.62)	0.98 (0.71)
f4	693.01 (514.62)	935.28 (803.64)	396401.4 (816218.3)	1311.66 (1313.8)	13309.6 (11892.7)	**637.7 (492.37)**	878.79 (941.35)
f5	567.1 (365.39)	29091.6 (72612.5)	8295.47 (9580.77)	653.97 (600.8)	1414.77 (870.2)	663.48 (513.14)	**483.25 (504.22)**
f6	4.89 (3.13)	12.16 (4.37)	1132.42 (560.95)	13.17 (19.36)	-Inf (NaN)	63.76 (290.53)	**1.81 (1.29)**
f7	12.95 (9.07)	16.39 (19.29)	480.18 (461.55)	39.19 (42.19)	203.29 (122.33)	**9.16 (21.73)**	18.09 (25.52)
f8	**3.22 (1.82)**	21180.05 (81706.26)	1272.34 (1632.38)	5.85 (5.83)	1.3E+7 (4165757)	3.6 (2.32)	3.77 (3.24)
f9	**489.83 (374.88)**	1116.77 (817.78)	447771.1 (524201.5)	1007.97 (1097.42)	12240.03 (10847.48)	767.45 (633.46)	820.4 (554.4)
f10	**481.6 (340.13)**	988.96 (789.75)	364702.1 (646525.6)	1884.76 (1862.31)	13838.06 (13264.11)	683.18 (555.79)	859.26 (701.19)
f11	3798.69 (1723.36)	2557.4 (726.55)	199352.7 (103017.5)	**1158.98 (1603.11)**	-Inf (NaN)	6190.25 (22872.84)	1224.63 (615.88)
f12	**7356.85 (6992.45)**	949080.5 (2658756)	1337235 (1573365)	23714.99 (23516.33)	68306.47 (50963.12)	8718.38 (6395.61)	14794.19 (11884.22)
f13	1447.36 (531.32)	673.28 (470.48)	5187.5 (840.21)	2294.52 (1132.65)	-Inf (NaN)	1717.61 (700.24)	**71.55 (74.76)**
f14	44485.7 (99044.43)	11629560 (27726330)	11325578 (12797968)	89687.58 (117749.4)	200089 (440769.7)	2688028 (85418.22)	**21795.63 (38171.18)**
f15	7761.61 (827.05)	3825.27 (2034.01)	12406.24 (1490.77)	7864.92 (1800.02)	-Inf (NaN)	7960.73 (851.99)	**436.98 (225.25)**
f16	**43085.64 (12055.25)**	565169.1 (92621.04)	43490751 (7726114)	406480.5 (113993.4)	800862 (66441.95)	358610.5 (55840.78)	1886637 (195888.5)
f17	**36837.62 (12768.45)**	542032.4 (71620.05)	38071684 (9666252)	376178.2 (85330.54)	778965.3 (47161.57)	391808.3 (68872.64)	1860255 (186114.9)
f18	**5289.22 (1337.58)**	266409.2 (216279.7)	6114410 (1743323)	55020.04 (18351.08)	138544.5 (13354.11)	47684.41 (7755.81)	250514.3 (26572.02)
f19	**31.95 (9.52)**	5.75E+8 (2.04E+9)	5217.11 (6576.03)	591.59 (206.81)	1565.76 (211.79)	455.89 (96.56)	2226.3 (411.5)
f20	1.65E+10 (6.32E+10)	5.69E+42 (3.12E+43)	208182 (39325.78)	1.98E+13 (4.19E+13)	**558.28 (357.76)**	6.76E+12 (1.92E+13)	1.87E+13 (4.83E+13)
f21	**534889.2 (167528.9)**	8333270 (1434724)	5.93E+08 (1.36E+08)	5529166 (1650880)	8762013 (814621.5)	4965894 (807743.2)	24789049 (2553670)
f22	**65038247 (26701308)**	4.1E+9 (9.9E+9)	3.66E+10 (1.35E+10)	1.11E+9 (3.76E+8)	6.81E+8 (87650243)	9.02E+8 (2.62E+8)	4.48E+9 (7.24E+8)
f23	**444674.9 (159837.9)**	4.33E+8 (2.28E+9)	2.99E+8 (1.07E+8)	8050411 (3345680)	8907776 (1398584)	6648936 (1992828)	30721396 (6114181)
f24	**1752.8 (695.73)**	110187.2 (21394.4)	610470.5 (187387.7)	15655.02 (12571.14)	292770.3 (102077.1)	15982.06 (2833.89)	146740.9 (12483.67)
f25	**361.38 (146.37)**	1221.29 (171.02)	900520 (154301.6)	3555.66 (1040.44)	10846.48 (703.09)	3486.91 (669.1)	17753.05 (1343.65)
f26	**143.13 (22.45)**	10421.61 (1730.37)	39269.29 (225.88)	408.51 (134.79)	1773.67 (103.38)	444.4 (33.21)	2359.89 (128.27)

the curve (for a given algorithm) with coordinates, (global min, initial best value) and (number of iterations, final best solution). In the example of Fig. 1, the global minimum is at 0 and the number of iterations is 2000, since we noticed no change in convergence behavior of the algorithm above 2000 runs. Each algorithm starts with an initial arbitrary value and ends with its final best value. The main advantage of using this measure is that it allows to get an insight of how fast the algorithm can converge towards the global minima as it incorporates the essence of both time taken (iterations) and the solution obtained. Table II shows the average AUC for 30 runs for different optimization algorithms tested over 26 benchmark functions. It is to be noted that the proposed method is better than the existing algorithms as it converges faster which is evident from the lesser AUC values obtained. The proposed algorithm is the best across 15 out of the 26 functions while the next best is 6 out of 26. The problem of searching the global solution becomes

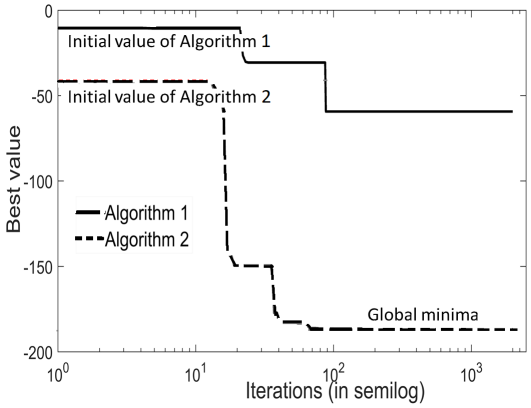

Fig. 1. Illustration of AUC measure for two algorithms for 2000 iterations. The area covered by each of the curve from their respective upper limits till the global minima is considered.

harder with the increase in the dimension of the variables in the test functions. In this study, the test functions from $f14$ to $f26$ have higher dimensions and it is seen that the proposed algorithm outperforms the rest in 10 out of 13 cases.

A. Use case - Eye tracking noise removal

Eye tracking finds various applications in cognitive state and well-being analysis [11], [12]. Obtaining noise free gaze data is vital for such applications [13]. Fig. 2 shows a sample gaze map obtained using an eye tracker against known target ground truth locations (shown as '+' sign). The corresponding estimations provided by the eye-tracker (shown using circular points) in the figure have an offset from the ground truth locations. This offset is considered as systematic error in the realm of eye tracking [13]. Systematic error is caused mainly due to badly performed calibration phase, and/or due to head movements of a subject and/or sudden jerk in the position of eye tracker after the initial calibration. Consider a cognitive test like the digit symbol substitution test (DSST) which involves eye tracking for assessing cognitive functions [11]. Gaze analysis is used to test which entities the participant is currently gazing at or the gaze path traversed. Due to systematic errors, the system can flag a trial as incorrect even if performed correctly. Also, in applications such as virtual keyboard control using eye tracking [14], systematic errors can result in activating unattended keys leading to several errors in terms of application control and adverse user experience. Hence, removal of systematic error is vital in eye gaze-based applications. We use this use case to show how the proposed algorithm can correct the systematic error significantly. Standard infrared-based eye trackers, generally comes with their own calibration phases, termed as primary calibration. However, even after properly undergoing this phase, it is seen that the systematic error still persists [13]. As the calibration phase is part of the hardware, it is not possible to get direct access to the learned parameters/models from the calibration phase, and thus we cannot fine tune the model to have further improvements in gaze detection. Hence, it is advisable to have another calibration phase (secondary) on top of the primary, with known target calibration points (9 point scheme as shown with '+' sign in Fig. 2). This secondary calibration can be deployed as follows. The task is

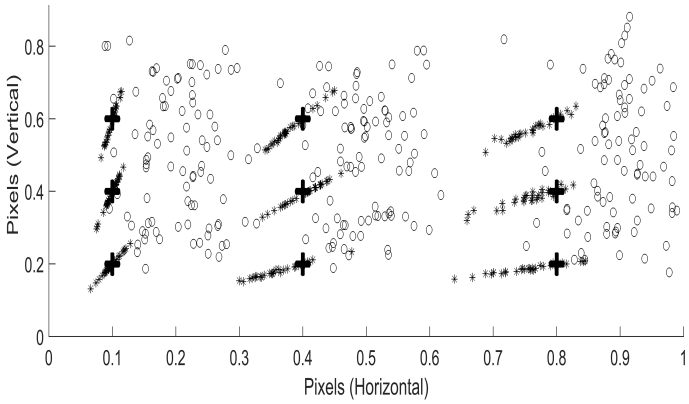

Fig. 2. Illustration of gaze points on the 9 static locations (fixation cross '+'). Circles indicate the raw eye tracker gaze points and the asterisk (*) shows the corrected points

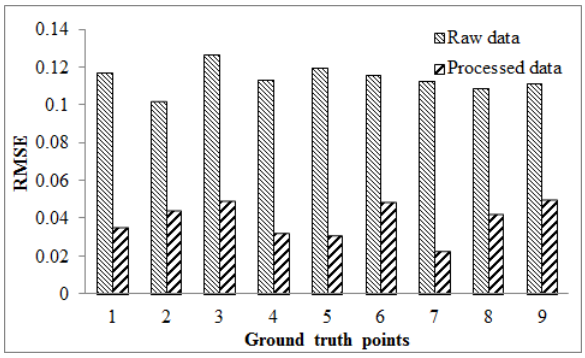

Fig. 3. RMSE values for the eye tracker provided and corrected gaze coordinates between the detected and the target ground truth locations.

to solve a minimization problem with the objective function being $|\overrightarrow{G} - \overrightarrow{C}|$, wherein $\overrightarrow{G}$ represents ground truth, in the calibration phase (here, secondary calibration). $\overrightarrow{C}$ represents corrected gaze coordinates, associated with raw gaze coordinates $\overrightarrow{F}$, received from an eye tracker and $|\overrightarrow{G} - \overrightarrow{C}|$ is the Euclidean distance between $\overrightarrow{G}$ and $\overrightarrow{C}$. Let $\overrightarrow{F}$ represent the raw gaze coordinates received from the eye tracker for a static position viewed on a screen. $\overrightarrow{F}$ is a 2-dimensional vector consisting of the horizontal and vertical screen coordinates, $\overrightarrow{F} = [(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)]$ where N is the number of points and is a function of the sampling rate of the eye tracker. The corrected gaze coordinates which is desired to be bereft of the systematic error may be obtained as, $\overrightarrow{C} = \overrightarrow{F} * T$, where T is a transformation matrix which needs to be derived for a static calibration point. The normalized screen coordinates are obtained when a subject is gazing at 9 different locations on the screen. The optimal transformation matrices for each of these 9 points are computed using the ground truth locations, in the calibration phase. The test phase makes use of these transformation matrices to correct the eye tracker data. Fig. 2 illustrates corrected gaze coordinates for the eye tracker data (as asterisk) in the test phase. It is noted that the processed data (asterisks) come closer to the ground truth locations (black fixation cross) in comparison to the raw eye tracker data (circles). The root mean square error (RMSE) between the gaze coordinates and the ground truth locations for both the eye tracker provided and corrected data (in the test phase) is shown in Fig. 3. It is to be noted that the usage of optimization helps in reducing the systematic error considerably.

IV. Conclusions and Future Scope

This study aimed at designing a new optimization algorithm motivated by the 3 important characteristics of caterpillar's food search behaviour, i.e. head waving (periscopic), usage of pheromone and movement type (fractality). The technique is tested on 26 standard test functions of varying difficulty levels. It is seen that the proposed algorithm is better in performance when compared to the closely related approaches. Finally, the optimization scheme is used to minimize the systematic error in eye tracking data. As a future roadmap, we would also like to incorporate the proposed eye tracking systematic error correction as primary calibration phase for in-house developed eye trackers using web camera.

References

[1] Shan He, Henry, and JR, "Group search optimizer: an optimization algorithm inspired by animal searching behavior," *IEEE transactions on evolutionary computation*, vol. 13, no. 5, pp. 973–990, 2009.

[2] Ramin Rajabioun, "Cuckoo optimization algorithm," *Applied soft computing*, vol. 11, no. 8, pp. 5508–5518, 2011.

[3] Swagatam Das and P Suganthan, "Differential evolution: A survey of the state-of-the-art," *ITEC*, vol. 15, no. 1, pp. 4–31, 2010.

[4] Yang and Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering computations*, 2012.

[5] Mirjalili, Mohammad, and Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.

[6] Seyedali and Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[7] Alireza Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm," *Computers & Structures*, vol. 169, pp. 1–12, 2016.

[8] Arora and Singh, "Butterfly optimization algorithm: a novel approach for global optimization," *Soft Computing*, pp. 715–734, 2019.

[9] SB Weiss and DD Murphy, "Fractal geometry and caterpillar dispersal: or how many inches can inchworms inch?," 1988.

[10] Michael J Katz, "Fractals and the analysis of waveforms," *Computers in biology and medicine*, vol. 18, no. 3, pp. 145–156, 1988.

[11] Chatterjee et al, "Evaluating age-related variations of gaze behavior for a novel digitized-digit symbol substitution test," *Journal of Eye Movement Research*, vol. 12, no. 1, 2019.

[12] Gavas et al, "Effect of cognitive load on a random sequence generation task," in *2019 IEEE Region 10 Symposium (TENSYMP)*. IEEE, 2019, pp. 243–248.

[13] Gavas et al, "Enhancing the usability of low-cost eye trackers for rehabilitation applications," *PloS one*, vol. 13, no. 6, 2018.

[14] Khasnobish et al, "Eyeassist: A communication aid through gaze tracking for patients with neuro-motor disabilities," in *PerCom Workshops*. IEEE, 2017, pp. 382–387.