European Journal of Technique

journal homepage: https://dergipark.org.tr/en/pub/ejt

Vol. 13, No. 2, 2023

Research Article

Performance Evaluation of Capuchin Search Algorithm Through Non-linear Problems, and Optimization

Erdal EKER[1*]

[1*]Muş Alparslan University, Vocational School of Social Science, Muş, Turkey. (e-mail: e.eker@alparslan.edu.tr).

ARTICLE INFO

Received: Nov., 15. 2023
Revised: Dec., 11. 2023
Accepted: Dec., 21. 2023

Keywords:
Metaheuristic algorithm
Optimization
CapSA
Gear train design problem

Corresponding author: *Erdal EKER*

ISSN: *2536-5010* | e-ISSN: *2536-5134*

DOI: https://doi.org/10.36222/ejt.1391524

ABSTRACT

The purpose of this paper is to demonstrate the superiority of the Capuchin Search Algorithm (CapSA), a metaheuristic, in competitive environments and its advantages in optimizing engineering design problems. To achieve this, the CEC 2019 function set was used. Due to the challenging characteristics of the CEC 2019 function set in reaching a global solution, it effectively showcases the algorithm's quality. For this comparison, sea-horse optimizer (SHO), grey wolf optimizer (GWO), sine-cosine algorithm (SCA), and smell agent optimization (SAO) were chosen as current and effective alternatives to the CapSA algorithm. Furthermore, the gear train design problem (GTD) was selected as an engineering design problem. In addition to the CapSA algorithm, a hybrid of SCA and GWO algorithm (SC-GWO) and genetic algorithm (GA) were chosen as alternatives for optimizing this problem. The performance superiority and optimization power of the CapSA algorithm were assessed using statistical metrics and convergence curves, then compared with alternative algorithms. Experimental results conclusively demonstrate the significant effectiveness and advantages of the CapSA algorithm.

1. INTRODUCTION

Real-world problems, in general, can be expressed as complex non-linear programming problems. Due to their combinatorial nature, these problems have been described as hard problems [1]. The structure of the objective functions at the core of these problems also determines the optimization process leading to the solution. Non-linear programming (NLP) is a mathematical programming technique in which the objective function is non-linear or one or more of the constraints have a non-linear relationship. NLP problems can be modeled in Eq. 1 given below [1]. Here, the objective function n is the number of variables, g and h are the constraints.

$$\left.\begin{array}{r} f_{min}(x) = f(x_1, x_2, x_3, \dots x_n) \\ g_i(x) \leq 0, i = 1,2,\dots,n \\ h_j(x) \leq 0, j = 1,2,\dots,n \end{array}\right\} \quad (1)$$

Traditional methods exist in the literature for solving the NLP problem. These methods, defined as gradient optimization techniques, attempt to solve these problems by using special mathematical structures and formulations [1]. Examples of these methods include the sequential unconstrained minimization technique [2], the augmented lagrangian [3], Newton-Raphson [4], the successive quadratic programming algorithm [5], the steepest descent algorithm [6], dynamic integer programming [7], and the stochastic Newton optimization method [8] as optimization techniques. The disadvantage of these techniques is that they are not suitable for solving complex optimization problems. Especially as the complexity of the problem increases with the addition of uncertainties to the system, more complex optimization techniques that overcome the limitations of classical approaches should be used. Metaheuristics have been developed with this goal in mind [9, 10]. Since metaheuristic algorithms do not use the derivative or second derivative of the objective function, they produce solutions in the neighborhood of the optimal solution. They avoid getting stuck in the local search space, reach the global solution in less time, and can hybridize with other algorithms to deal with different problems, having a flexible structure with all these features [11-13]. Due to their repetitive working methods and memory utilization, they can make new explorations without having to go back to the beginning each time [14]. Metaheuristic algorithms consist of three different mechanisms: the initialization phase, containing the candidate solution set; the exploitation phase, which divides the search space into regions to concentrate on narrow areas; and the exploration phase, which scans the entire space, selects, and improves the best solutions obtained in the exploitation phase. The success of metaheuristic algorithms is directly proportional to the strength of the balance between the exploitation and exploration phases [15]. The CapSA algorithm considered in this paper is one of the current and efficient metaheuristic algorithms produced in 2021 [16].

Many studies on CapSA can be mentioned in the literature. The main ones are explained as follows. In the first paper introducing the algorithm, Braik et all experienced the optimization power of the CapSA algorithm in classic engineering design problems such as welded beam design, pressure vessel design, tension-compression spring design, speed reducer design [16]. Braik, one of the inventors of the CapSA algorithm, proposed the MGP-CapSA algorithm by combining the CapSA algorithm with the multigene genetic algorithm, and with this algorithm, he produces a new simulator model for the wrapping problem from non-linear problems and uses CapSA to optimize the coefficients of the regression equations of MGP [17]. Kanipariya et all produced the ICSA algorithm by hybridising the CapSA algorithm with adversarial learning and chaotic local search algorithms and used it to classify lung nodule abnormalities [18]. Fathy et all used the CapSA algorithm to minimize grid active power loss by maintaining power flow, bus voltage and transmission line within their normal ranges in electricity distribution networks [19]. A similar study was conducted by Zakaria et all [20]. Ramu et all used the modified CapSA algorithm (MCS) to solve the cloud performance scheduling problem, which minimizes the completion time and improves resource utilisation [21]. Broumandnia et all hybridized the CapSA algorithm with the inverted ant colony optimization (IACO) algorithm for the optimization of some processes related to the cloud system and obtained better performance [22]. Qin et all, CapSA based PID control system for solving industrial problems [23]. Kumar et all used CapSA algorithm as an optimizer to classify normal and malicious attacks by strengthening the security scheme of IoT (Internet of Things) [24]. In a similar study, Rani and Burty hybridized CapSA with different machine learning algorithms for smart home energy management [25]. Ehteram et all used the CapSA algorithm as a basis for training ANNs to carry out evaporation prediction [26]. Alphonse et all used the ECapSA technique, a combination of CapSA and wild horse optimizer, to enable the simultaneous allocation of electric vehicle charging station (EVCS) and photovoltaic (PV) energy sources in the smart grid [27]. The gear train design problem, which is optimized in this study, has been studied by many metaheuristic algorithms before and effective results have been obtained [28-32].

The organization of the paper is as follows. Section 1 consists of the introduction and the optimization process of the study. The second section consists of the introduction and mathematical modelling of the CapSA algorithm. Experimental studies are included in the third section. The sub-sections of this section consist of the introduction of the CEC 2019 function set and the comparison of the CapSA algorithm with alternative algorithms through statistical results and observing the performance superiority of the CapSA algorithm through tables and converge curves. The other subsection includes the optimization of the Gear train design problem with the help of CapSA and alternative algorithms, The performance results are shown through tables and convergence curves, and the fourth section contains the conclusions of the study.

2. CAPUCHIN SEARCH ALGORITHM (CapSA)

The CapSA algorithm, based on swarm intelligence, was created by simulating the foraging behavior of Capuchin monkeys living in the Americas, with a particular emphasis on their jumping abilities [16, 33]. The mathematical model of the CapSA algorithm is constructed to align the jumping movements of Capuchin monkeys with the exploration phase, while the swinging and climbing movements correspond to the exploitation phase. In the first stage, a random initial set of a specific number of candidates is created [16]. The initial set is represented by a matrix of size *dxn.*

$$x = \begin{bmatrix} x_1^1 & \dots & x_d^1 \\ \vdots & \dots & \vdots \\ x_1^n & \dots & x_d^n \end{bmatrix}_{d\times n} \quad (2)$$

The initial location of each Capuchin monkey is expressed in Eq. 3. Here ub_j and lb_j, denotes the upper and lower boundaries of the *i.* monkey in the *jth* dimension, respectively operator r corresponds to a
random number uniformly distributed in the closed interval [0,1].

$$x^j = ub_j + r \times (ub_j - lb_j) \quad (3)$$

The position of the leader monkey during the tree climbing behavior is modelled in Eq. 4. Here, the *i.* positions of the leader and accompanying monkeys in the *j.* dimension x_j^i, F_j the location of the food,
v_j^i the speed of the monkey, P_{bf}, the probability of balance provided by the monkey's tail throughout the jumping movement, The gravitational force, g, corresponding to the number 9.81, ε operator expresses a random number informly distributed in the interval [0,1].

$$\left.\begin{array}{l} x_j^i = F_j + \frac{P_{bf}\left(v_j^i\right)^2 sin(2\pi)}{g} \\ i < \frac{n}{2},\ 0.10 < \varepsilon \le 0.20 \end{array}\right\} \quad (4)$$

Monkeys jump angle (θ) is given in Eq. 5. Here operator r expresses a random number irregularly distributed in the interval [0,1].

$$\theta = \frac{3}{2} r \quad (5)$$

In Eq. 6, we show that CapSA is a system that updates the monkeys' locations to quickly detect the location of the food source by exploring and exploiting the search space. τ operator is defined.

$$\tau = \beta_0 e^{\beta_1 \left(\frac{t}{T}\right)^{\beta_2}} \quad (6)$$

where t and T, denote the current iteration and maximum iteration, respectively, β_0, β_1, β_2 it is stated that after many experimental studies, the author has chosen 2, 21 and 2 as the most appropriate values for the parameters, respectively. The appropriate value of the parameter τ strengthens the exploration and exploitation capabilities of the CapSA algorithm.

In Eq.7, the speed of monkey i. in the jth dimension is modelled. Here, the velocity of monkey ith in the *jth* dimension v_j^i, their location x_j^i, best location $x_{best_j}^i$ refers to r_1 ve r_2 is a random number uniformly in the range [0,1]. ρ, coefficient of inertia controls the effect of the previous velocity on the motion. In this equation, there are also two fixed control parameters namely a_1 and a_2 that control the speed of the monkeys by adjusting the parameters $x_{best_j}^i$and F_j. ρ, a_1 and a_2 are parameters arbitrary.

$$v_j^i = \rho v_j^i + \tau a_1 \left(x_{best_j}^i - x_j^i \right) r_1 + \tau a_2 (F_j - x_j^i) r_2 \quad (7)$$

The new position of the leader and the accompanying monkeys as a result of the jump is modelled in Eq.8

$$x_j^i = F_j + \frac{P_{ef} P_{bf} (v_j^i) \sin(2\theta)}{g}, i < \frac{n}{2}, 0.20 < e \leq 0.30 \quad (8)$$

given P_{ef} is the probability of the monkey yawning. The new position of the Alpha monkey is modelled in Eq. 10. Given by $P_{bf} = 0.7$ and $P_{ef} = 9$.

$$x_j^i = x_j^i + v_j^i,\ i < \frac{n}{2}, 0.20 < e \leq 0.30 \quad (9)$$

Local foraging is achieved by the swaying movement that alpha and its companion monkeys use to collect food. The positions of the monkeys in this situation are formulated below.

$$x_j^i = F_j + \tau P_{bf} \times sin(2\theta), i < \frac{n}{2}, 0.50 < e \leq 0.75 \quad (10)$$

Similar to local foraging, the leader and its companion monkeys may repeat the behavior of foraging repeatedly. The positions of monkeys displaying this behavior are modelled following.

$$x_j^i = F_j + \tau P_{bf} \left(v_j^i - v_{j-1}^i \right) i < \frac{n}{2}, 0.75 < e \leq 1.00 \quad (11)$$

where the ith monkey in the *jth* dimension v_j^i current speed, v_{j-1}^i indicates the previous speed.

Monkeys can also move randomly to find food. This is given in the equation below. In the equation P_r is equal to 0 and expresses the random search probability of the monkeys. The randomization capability and the monkey herd behavior expressed here improves the global search capability of the algorithm and supports the escape from local optimum points. τ parameter has the role of strengthening the equilibrium position between exploration and exploitation.

$$x_j^i = \tau \times \left(lb_j + e \times \left(ub_j - lb_j \right) \right), i < \frac{n}{2}, e \leq P_r \quad (12)$$

Eq. 13 models the leader monkey updating the position of its followers based on the third law of motion.

$$x_f = x_i + v_0 t + \frac{1}{2} a t^2 \quad (13)$$

$$a = \frac{\Delta v}{\Delta t} = \frac{v_f - v_0}{t_1 - t_0} \quad (14)$$

$$v_f = \frac{\Delta x}{\Delta t} = \frac{x_f - x_0}{t_1 - t_0} \quad (15)$$

given in Eq.13 x_f and x_i with the displacements in the initial and final phases t time, v_0 initial speed, a is the slope whose formula is given in Eq. 14. v_f final speed, v_0 initial velocity, respectively $t_1$ve t_0 give the final and start times. In Eq. 16 $v_0 = 0$ when taken a becomes as formulated in Eq.16.

$$a = \frac{x_f - x_0}{(t_1 - t_0)^2} \quad (16)$$

Since Capuchin monkeys live in herds, it is important to simulate the behavior of the leader as well as the behavior of the followers as they follow the leader.

$$x_j^i = \frac{1}{2} \left(x'^i_j + x_j^{i-1} \right), \frac{n}{2} \leq i \leq n \quad (17)$$

From the expressions given in Eq.17, x_j^i current position of the followers in dimension *jth*, x_j^{i-1} previous location, x'^i_j is the current position of the leader. Since the time intervals in the simulation refer to iterations $t_i - t_{i-1} = 1$.

The fitness function for each monkey is evaluated by adjusting the solution vector values in a fitness function and stored in a matrix as expressed in Eq.18, which serves as a memory.

$$f = \begin{bmatrix} f_1([x_1^1, x_2^1, \ldots, x_d^1]) \\ \vdots \quad \vdots \quad \vdots \\ f_n([x_1^n, x_2^n, \ldots, x_d^n]) \end{bmatrix}_{d \times n} \quad (18)$$

3. EXPERIMENTAL RESULTS

The CEC2019 function set was employed to assess the optimization capability of the CapSA algorithm. This set consists of difficult problems. The goal of these problems is to highlight the optimization capability and competitive aspect of the algorithm by compelling it to reach the global solution [15]. For the alternative algorithms that CapSA will compete with, we selected current and efficient algorithms. These include the grey wolf optimizer (GWO) [34], sea-horse optimizer (SHO) [35], sine-cosine algorithm (SCA) [36], and smell agent optimization (SAO) [37]. When evaluating the performance of the algorithm, we conducted 30 independent runs with 500 iterations and 30 search agents in each run. In this study, CapSA parameters took arbitrary values ρ, a_1 and a_2 which were set to 0.7, 1, and 1 respectively.

TABLE I
CEC 2019 FUNCTIONS

Functions	Dimension	[Lower&Upper bound]	Fit. V
Function 1	9	[−8192, 8192]	1
Function 2	16	[−16384, 16384]	1
Function 3	18	[-4,4]	1
Function 4	10	[−100, 100]	1
Function 5	10	[−100, 100]	1

Function 6	10	[−100, 100]	1
Function 7	10	[−100, 100]	1
Function 8	10	[−100, 100]	1
Function 9	10	[−100, 100]	1
Function 10	10	[−100, 100]	1

In Table 1, the optimal values of CEC 2019 are provided. As the optimal value for all functions is 1, the algorithm observed to be the most effective experimentally is expected to be closer to 1 than the others. For this reason, optimal results are expected from both the best value and the average value. The difference between the best value and the worst value should not be too significant, indicating that the standard deviation value should be low.

TABLE 2
. PERFORMANCE OF CEC2019

CEC2019 Functions	Metrics	Algorithms				
		CapSA	SHO	GWO	SCA	SAO
Function 1	Mean	**4.1043E+04**	4.5384E+04	1.3297E+08	8.5744E+09	3.3710E+11
	Std.dev.	2.6591E+03	**2.5926E+03**	2.4538E+08	9.4139E+09	4.2532E+11
	Best	**3.7079E+04**	4.0869E+04	9.0425E+04	4.5544E+07	1.0675E+06
	Worst	**4.8634E+04**	5.0757E+04	1.0915E+09	4.5484E+10	1.5831E+12
	Run time (sec.)	6.2867	9.4452	6.2162	**6.1441**	24.0524
	Rank	**1**	2	3	4	5
Function 2	Mean	**1.7342E+01**	1.7386E+01	1.7344E+01	1.7485E+01	2.1755E+03
	Std.dev.	**6.4444E-05**	1.0166E-01	3.4780E-04	6.6712E-02	2.1538E+03
	Best	**1.7342E+01**	1.7343E+01	1.7343E+01	1.7397E+01	1.7881E+01
	Worst	**1.7343E+01**	1.7677E+01	1.7345E+01	1.7710E+01	8.3069E+03
	Run time (sec.)	**0.2711**	0.8144	0.3256	0.3926	0.6813
	Rank	**1**	3	2	4	5
Function 3	Mean	**1.2702E+01**	1.2702E+01	1.2702E+01	1.2702E+01	1.2703E+01
	Std.dev.	**9.0336E-15**	4.8734E-06	3.4200E-04	9.4149E-05	8.5018E-04
	Best	**1.2702E+01**	1.2702E+01	1.2702E+01	1.2702E+01	1.2702E+01
	Worst	**1.2702E+01**	1.2702E+01	1.2704E+01	1.2703E+01	1.2706E+01
	Run time (sec.)	**0.3935**	0.8796	0.4922	0.4089	0.8863
	Rank	**1**	2	4	3	5
Function 4	Mean	**4.3731E+01**	1.4375E+03	6.1384E+01	1.5444E+03	1.6066E+04
	Std.dev.	**1.9984E+01**	1.5552E+03	2.2991E+01	5.5463E+02	5.1770E+03
	Best	**1.0950E+01**	9.5000E+01	2.0309E+01	6.3374E+02	7.1411E+03
	Worst	**8.6579E+01**	4.7645E+03	1.0763E+02	2.6895E+03	2.8970E+04
	Run time (sec.)	0.3186	0.7399	0.3504	**0.2304**	0.79839
	Rank	**1**	3	2	4	5
Function 5	Mean	**1.3192E+00**	1.7793E+00	1.4540E+00	2.2394E+00	5.3936E+00
	Std.dev.	1.7340E-01	2.5219E-01	2.5398E-01	**1.2223E-01**	1.2545E+00
	Best	**1.0689E+00**	1.2928E+00	1.0790E+00	2.0473E+00	2.8281E+00
	Worst	**1.6598E+00**	2.4198E+00	1.8360E+00	2.6946E+00	8.2563E+00
	Run time (sec.)	**0.1385**	0.4208	0.1983	0.1867	0.6286
	Rank	**1**	3	2	4	5
Function 6	Mean	**7.9079E+00**	8.0011E+00	1.1066E+01	1.0843E+01	1.0121E+01
	Std.dev.	1.3018E+00	9.9758E-01	7.0067E-01	6.3186E-01	6.9379E-01
	Best	**4.8923E+00**	5.7081E+00	9.7446E+00	9.3411E+00	8.9621E+00
	Worst	1.0783E+01	**1.0004E+01**	1.2501E+01	1.1967E+01	1.1699E+01
	Run time (sec.)	2.3342	3.5275	**2.2171**	2.2328	10.2188
	Rank	**1**	2	5	4	3
Function 7	Mean	4.3136E+02	**2.8850E+02**	4.3643E+02	7.7376E+02	1.1062E+03
	Std.dev.	3.2594E+02	**1.0717E+02**	3.0061E+02	1.7682E+02	3.9157E+02
	Best	**-6.3737E+01**	-1.4686E+01	5.8673E+01	3.9072E+02	3.7941E+02
	Worst	1.3055E+03	**5.3524E+02**	1.2014E+03	1.0398E+03	2.2533E+03
	Run time (sec.)	**0.1595**	0.4134	0.1704	0.1974	0.5529
	Rank	2	**1**	3	4	5
Function 8	Mean	5.4122E+00	5.4523E+00	**5.2877E+00**	6.1099E+00	6.4154E+00
	Std.dev.	6.899E-01	5.4020E-01	9.0214E-01	4.5161E-01	**3.4621E-01**
	Best	3.9460E+00	4.4703E+00	**3.5009E+00**	5.0269E+00	5.7638E+00
	Worst	**6.6099E+00**	6.3853E+00	6.9706E+00	6.8575E+00	7.2611E+00
	Run time (sec.)	**0.1443**	0.4205	0.2034	0.1816	0.6881
	Rank	2	3	1	4	5
Function 9	Mean	**2.7197E+00**	1.5525E+02	4.4281E+00	1.4306E+02	2.6530E+03
	Std.dev.	**0.3344E-01**	2.7471E+02	9.2393E-01	1.1088E+02	8.8512E+02
	Best	**2.4323E+00**	4.4605E+00	2.5669E+00	3.2968E+00	6.0353E+02
	Worst	**4.1519E+00**	8.0767E+02	6.3641E+00	4.7436E+02	4.6614E+03
	Run time (sec.)	**0.1644**	0.4103	0.1806	0.1978	0.5266
	Rank	**1**	4	2	3	5

Function 10	Mean	**1.95338E+01**	2.0112E+01	2.0503E+01	2.0467E+01	2.0455E+01
	Std.dev.	2.9277E+00	7.8076E-02	8.1513E-02	9.1492E-02	1.6069E-01
	Best	**4.0361E+00**	1.9993E+01	2.0299E+01	2.0280E+01	2.0223E+01
	Worst	**2.0231E+01**	2.0280E+01	2.0648E+01	2.0632E+01	2.1142E+01
	Run time (sec.)	**0.1457**	0.4087	0.3611	0.3172	0.6346
	Rank	**1**	2	5	4	3

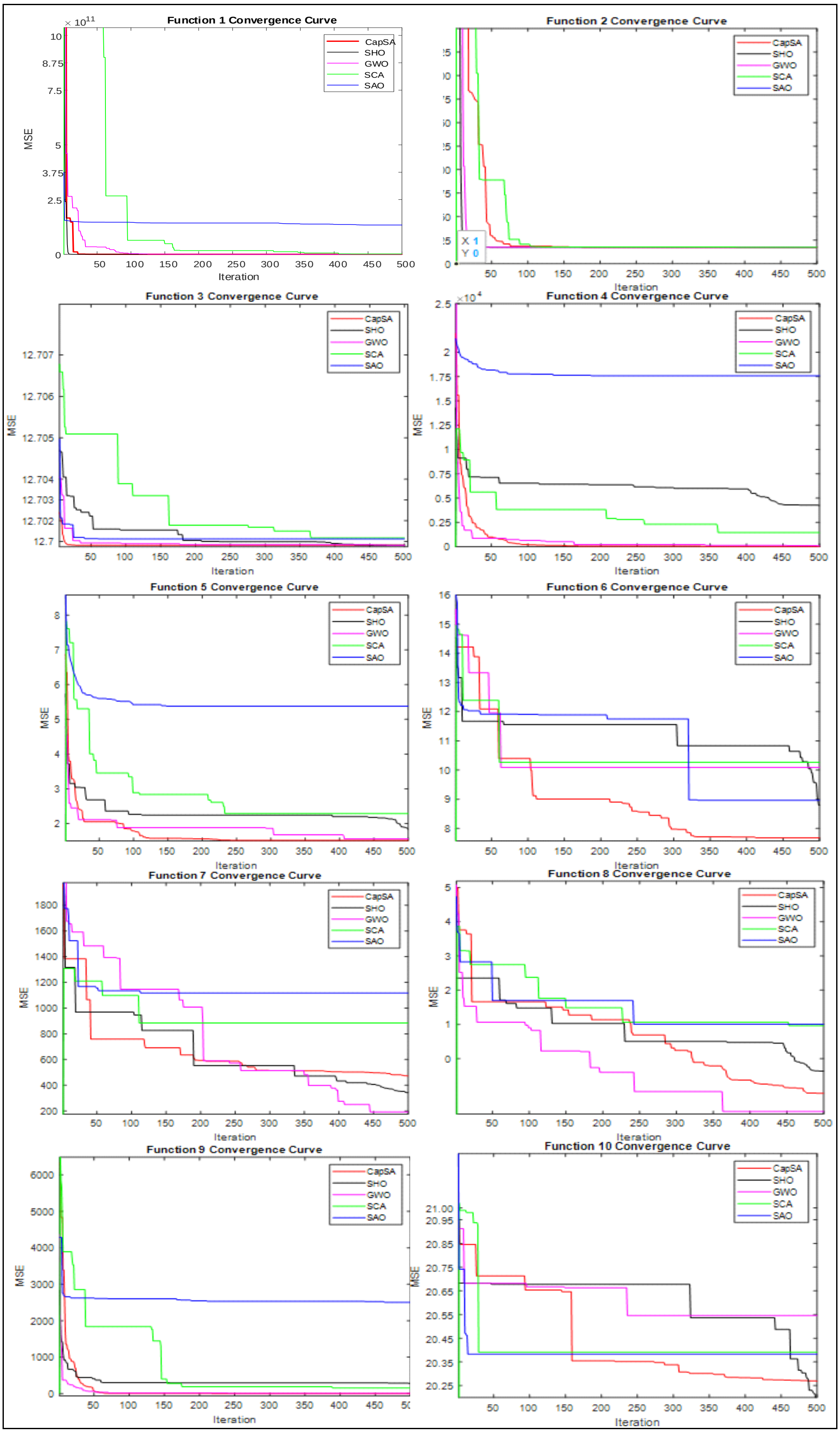

Figure 1. Convergence curve of algorithms via CEC 2019 function

In this context, upon analyzing Table 2, it is observed that the CapSA algorithm is more advantageous from function 1 to function 5. It is advantageous in terms of both average and best value in function 6, and it is superior in functions 9 and 10. While it completed the optimization process in the shortest time for almost all functions, it took slightly longer in functions 1, 4, and 6. When looking at the average values, except for functions 7 and 8, it has consistently held its status as the most superior algorithm. The CapSA algorithm was run for the first time with the CEC 2019 function set and it was clearly seen that it has a strong competitive aspect and a structure that can reach global results without getting stuck in the local area. Convergence curves are statistical measures that indicate whether algorithms are converging early or sticking to local optimum points as they iteratively progress through the process of optimizing a function. In the Figure1, when the CAPSA algorithm is compared with alternative algorithms, the algorithm is labelled as 1. It is observed that it cannot progress to the local optimum point in the function, but it progresses steadily towards the optimum point in the other functions for 500 iterations. In the 7th and 8th functions, GWO algorithm shows better convergence, while CapSA shows the best convergence in all other algorithms.

4. GEAR TRAIN DESİGN PROBLEM

Figure 2 shows the design of a gear train, which is set up to determine the number of teeth in each gear to produce a given speed ratio between the input and output shaft. Here A, B, C and D indicate the number of gears in each wheel. In order to minimize the ratio of angular velocity variation between input and output in accordance with the objective of the gear train design problem, a mathematical model is established by Eq. 19 [38-40].

$$\left.\begin{aligned} minf(x) &= \left(\frac{1}{6.931} - \frac{x_2 x_3}{x_1 x_4}\right)^2 \\ \vec{x} &= [A, B, C, D] = [x_1, x_2, x_3, x_4],\ \ 12 \le x_i \le 60 \end{aligned}\right\} \quad (19)$$

In Eq. 18, $\frac{x_2 x_3}{x_1 x_4}$ expression gives the gear ratio.

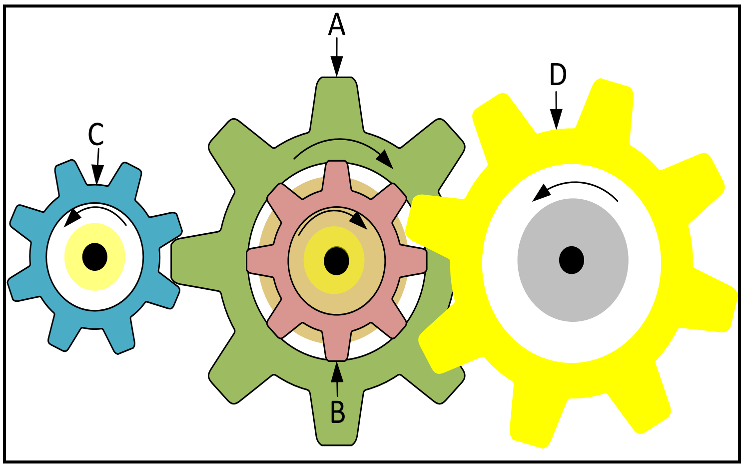

Figure 2. GTD model

TABLE 3
. GTD ANALYSIS RESULTS

	Parameters				Metrics				
Algorithm	x_1	x_2	x_3	x_4	Mean	Std. Dev.	Best	Worst	Rank
CapSA	**12**	**32**	**52**	**51**	**2.3987e-19**	**1.3080e-18**	**1.2326e-32**	**7.1655e-18**	**1**
GWO	15	22	55	45	5.2328e-12	1.0711e-11	1.3783e-15	4.5145e-11	4
SCA	15	29	60	51	1.5854e-09	2.2843e-09	6.9337e-13	7.4779e-09	5
SHO	12	12	48	21	7.4748e-09	1.7250e-08	7.5125e-17	7.1094e-08	6
SAO	21	58	58	18	3.0622e-03	1.6206e-02	2.0291e-17	8.8858e-02	7
SC-GWO[40]	43	16	19	49	2.7009E-12	-	-	-	3
GA[41]	19	16	43	49	2.7000E-12	-	-	-	2

The results of the performance comparisons of competitive metaheuristic algorithms help to determine the most effective algorithm in the optimization of the problem and the emergence of the optimum model. In this context, the analysis results of the CapSA, GWO, SCA, SHO, SAO, SC-GWO and GA algorithms for the GTD problem are shown and evaluated in Table 3. Here, the optimization results of SC-GWO algorithm, which is a hybrid algorithm of SCA and GWO, and Genetic algorithm (GA) for GTD problem are taken from previous studies. [40, 41].

When Table 3 is observed, the optimal result is found for the gear arrangement ratio generated by the CapSA algorithm. Here, the approximate number of gears and the synchronized encounter with each other reveal that it shows high performance compared to other algorithms. When Figure 3 is analyzed, it will be seen that the visual dimension of the table is parallel to the results in Table 3. In fact, in the convergence curve, it can be seen that the CapSA algorithm is quite stable and gets results around zero.

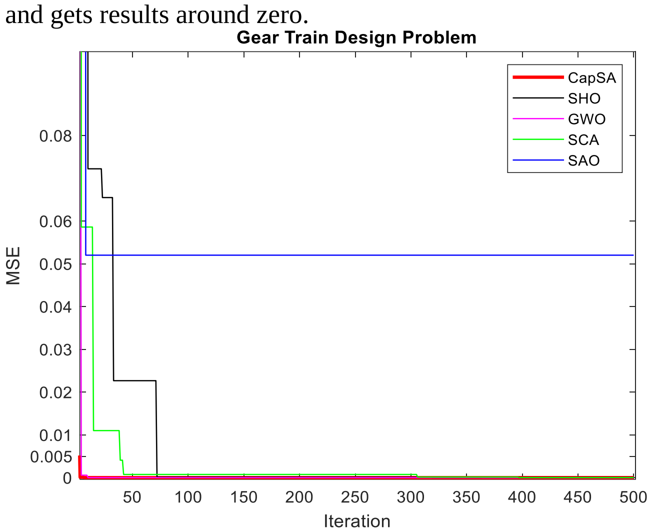

Figure 3. Convergence curve of GTD model via algorithms

5. DISCISSION

The aim of this paper is to reveal the competitiveness and performance superiority of the CapSA algorithm by comparing it with alternative algorithms through the CEC 2019 quality function set and to optimize the gear train design problem, which is one of the classic engineering problems. CapSA has experimentally shown that it is an algorithm with more advantageous results compared to alternative algorithms with various statistical measurements. Likewise, it has been experimentally observed that CapSA algorithm has the most optimal results in the optimization of GTD problem. Based on these results, it can be stated that the CapSA algorithm has a strong competitive structure, is stable in solving real world problems and has the flexibility to overcome the problems of getting stuck in the local algorithm in some functions seen in the structure of the algorithm when it is developed. For this reason, the CapSA algorithm is a promising algorithm that can be addressed in future studies with different aspects.

REFERENCES

[1] L. Costa ve P. Oliveira, “Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems”, *Computers & Chemical Engineering*, c. 25, sy 2-3, ss. 257-266, 2001.
[2] A. V. Fiacco ve G. P. McCormick, “The Sequential Unconstrained Minimization Technique for Nonlinear Programing, a Primal-Dual Method”, *Management Science*, c. 10, sy 2, ss. 360-366, Oca. 1964, doi: 10.1287/mnsc.10.2.360.
[3] H. Adeli ve N. Cheng, “Augmented Lagrangian Genetic Algorithm for Structural Optimization”, *J. Aerosp. Eng.*, c. 7, sy 1, ss. 104-118, Oca. 1994, doi: 10.1061/(ASCE)0893-1321(1994)7:1(104).
[4] A. Ben-Israel, “A Newton-Raphson method for the solution of systems of equations”, *Journal of Mathematical analysis and applications*, c. 15, sy 2, ss. 243-252, 1966.
[5] P. T. Boggs ve J. W. Tolle, “Sequential quadratic programming”, *Acta numerica*, c. 4, ss. 1-51, 1995.
[6] B. Widrow ve J. McCool, “A comparison of adaptive algorithms based on the methods of steepest descent and random search”, *IEEE transactions on antennas and propagation*, c. 24, sy 5, ss. 615-637, 1976.
[7] J. Zou, S. Ahmed, ve X. A. Sun, “Stochastic dual dynamic integer programming”, *Math. Program.*, c. 175, sy 1-2, ss. 461-502, May. 2019, doi: 10.1007/s10107-018-1249-5.
[8] B. Bullins, K. Patel, O. Shamir, N. Srebro, ve B. E. Woodworth, “A stochastic newton algorithm for distributed convex optimization”, *Advances in Neural Information Processing Systems*, c. 34, ss. 26818-26830, 2021.
[9] K. Sörensen ve F. Glover, “Metaheuristics”, *Encyclopedia of operations research and management science*, c. 62, ss. 960-970, 2013.
[10] S. E. De León-Aldaco, H. Calleja, ve J. A. Alquicira, “Metaheuristic optimization methods applied to power converters: A review”, *IEEE Transactions on Power Electronics*, c. 30, sy 12, ss. 6791-6803, 2015.
[11] D. H. Wolpert ve W. G. Macready, “No free lunch theorems for optimization”, *IEEE transactions on evolutionary computation*, c. 1, sy 1, ss. 67-82, 1997.
[12] E. Eker, M. Kayri, S. Ekinci, ve D. Izci, “A new fusion of ASO with SA algorithm and its applications to MLP training and DC motor speed control”, *Arabian Journal for Science and Engineering*, c. 46, ss. 3889-3911, 2021.
[13] E. Eker, M. Kayri, S. Ekinci, ve D. İzci, “Comparison of Swarm-based Metaheuristic and Gradient Descent-based Algorithms in Artificial Neural Network Training”, *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, c. 12, sy 1, ss. e29969-e29969, 2023.
[14] J. O. Agushaka, A. E. Ezugwu, L. Abualigah, S. K. Alharbi, ve H. A. E.-W. Khalifa, “Efficient initialization methods for population-based metaheuristic algorithms: a comparative study”, *Archives of Computational Methods in Engineering*, c. 30, sy 3, ss. 1727-1787, 2023.
[15] E. Eker, M. Kayri, S. Ekinci, ve M. A. Kaçmaz, “Performance Evaluation of PDO Algorithm through Benchmark Functions and MLP Training”, *Electrica*, c. 23, sy 3, 2023, Erişim: 04 Ekim 2023. [Çevrimiçi]. Erişim adresi: https://electricajournal.org/Content/files/sayilar/96/597-606.pdf
[16] M. Braik, A. Sheta, ve H. Al-Hiary, “A novel meta-heuristic search algorithm for solving optimization problems: capuchin search algorithm”, *Neural Comput & Applic*, c. 33, sy 7, ss. 2515-2547, Nis. 2021, doi: 10.1007/s00521-020-05145-6.
[17] M. Braik, “A hybrid multi-gene genetic programming with capuchin search algorithm for modeling a nonlinear challenge problem: Modeling industrial winding process, case study”, *Neural Processing Letters*, c. 53, sy 4, ss. 2873-2916, 2021.
[18] M. Kanipriya, C. Hemalatha, N. Sridevi, S. R. SriVidhya, ve S. J. Shabu, “An improved capuchin search algorithm optimized hybrid CNN-LSTM architecture for malignant lung nodule detection”, *Biomedical Signal Processing and Control*, c. 78, s. 103973, 2022.
[19] A. Fathy, D. Yousri, H. Rezk, ve H. S. Ramadan, “An efficient capuchin search algorithm for allocating the renewable based biomass distributed generators in radial distribution network”, *Sustainable Energy Technologies and Assessments*, c. 53, s. 102559, 2022.

[20] M. Zakaria, M. S. Seif, ve M. A. Mehanna, "Energy Management of MG Considering the Emission and Degradation Costs using A CAP-SA Optimization", *International Journal of Renewable Energy Research (IJRER)*, c. 12, sy 3, ss. 1452-1462, 2022.
[21] S. Ramu, R. Ranganathan, ve R. Ramamoorthy, "Capuchin search algorithm based task scheduling in cloud computing environment", *Yanbu Journal of Engineering and Science*, c. 19, sy 1, ss. 18-29, 2022.
[22] A. Broumandnia, S. Rostami, ve A. Khademzadeh, "An Energy-Efficient Task Scheduling Method for Heterogeneous Cloud Computing Systems Using Capuchin Search and Inverted Ant Colony Optimization Algorithms", *Available at SSRN 4270250*, Erişim: 13 Ekim 2023. [Çevrimiçi]. Erişim adresi: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4270250
[23] C. Qin, H. Hu, H. Chen, ve B. Yan, "Research on Optimization Control of Deep Hole Machining Based on Capuchin Search Algorithm to Optimize Fuzzy PID", içinde *2022 IEEE 21st International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS)*, IEEE, 2022, ss. 288-293. Erişim: 13 Ekim 2023. [Çevrimiçi]. Erişim adresi: https://ieeexplore.ieee.org/abstract/document/10086312/
[24] C. U. Om Kumar, J. Durairaj, S. A. Ahamed Ali, Y. Justindhas, ve S. Marappan, "Effective intrusion detection system for IoT using optimized capsule auto encoder model", *Concurrency and Computation*, c. 34, sy 13, s. e6918, Haz. 2022, doi: 10.1002/cpe.6918.
[25] V. U. Rani ve L. R. Burthi, "Power Quality Enhancement of Smart Home Energy Management System in Smart Grid Using MAORDF-CapSA Technique", *Ecological Engineering & Environmental Technology*, c. 23, 2022, Erişim: 13 Ekim 2023. [Çevrimiçi]. Erişim adresi: https://yadda.icm.edu.pl/yadda/element/bwmeta1.element.baztech-16773c9e-e170-4b5c-a9cf-242d6fe17d16
[26] M. Ehteram, F. Panahi, A. N. Ahmed, A. H. Mosavi, ve A. El-Shafie, "Inclusive multiple model using hybrid artificial neural networks for predicting evaporation", *Frontiers in Environmental Science*, c. 9, s. 789995, 2022.
[27] A. R. A. Alphonse, A. P. P. G. Raj, ve M. Arumugam, "Simultaneously allocating electric vehicle charging stations (EVCS) and photovoltaic (PV) energy resources in smart grid considering uncertainties: A hybrid technique", *Intl J of Energy Research*, c. 46, sy 11, ss. 14855-14876, Eyl. 2022, doi: 10.1002/er.8187.
[28] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, ve W. Zhao, "Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems", *Engineering Applications of Artificial Intelligence*, c. 114, s. 105082, 2022.
[29] H. T. Sadeeq ve A. M. Abdulazeez, "Giant trevally optimizer (GTO): A novel metaheuristic algorithm for global optimization and challenging engineering problems", *IEEE Access*, c. 10, ss. 121615-121640, 2022.
[30] N. Chopra ve M. M. Ansari, "Golden jackal optimization: A novel nature-inspired optimizer for engineering applications", *Expert Systems with Applications*, c. 198, s. 116924, 2022.
[31] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm", *Knowledge-based systems*, c. 89, ss. 228-249, 2015.
[32] S. Mirjalili, "The ant lion optimizer", *Advances in engineering software*, c. 83, ss. 80-98, 2015.
[33] V. F. Lenzen, "Newton's Third Law of Motion", *Isis*, c. 27, sy 2, ss. 258-260, Ağu. 1937, doi: 10.1086/347244.
[34] S. Mirjalili, S. M. Mirjalili, ve A. Lewis, "Grey wolf optimizer", *Advances in engineering software*, c. 69, ss. 46-61, 2014.
[35] S. Zhao, T. Zhang, S. Ma, ve M. Wang, "Sea-horse optimizer: a novel nature-inspired meta-heuristic for global optimization problems", *Applied Intelligence*, c. 53, sy 10, ss. 11833-11860, 2023.
[36] S. M. Mirjalili, S. Z. Mirjalili, S. Saremi, ve S. Mirjalili, "Sine cosine algorithm: theory, literature review, and application in designing bend photonic crystal waveguides", *Nature-inspired optimizers: theories, literature reviews and applications*, ss. 201-217, 2020.
[37] A. T. Salawudeen, M. B. Mu'azu, A. Yusuf, ve A. E. Adedokun, "A Novel Smell Agent Optimization (SAO): An extensive CEC study and engineering application", *Knowledge-Based Systems*, c. 232, s. 107486, 2021.
[38] M. Mendez, D. A. Rossit, B. González, ve M. Frutos, "Proposal and comparative study of evolutionary algorithms for optimum design of a gear system", *IEEE Access*, c. 8, ss. 3482-3497, 2019.
[39] E. Sandgren, "Nonlinear Integer and Discrete Programming in Mechanical Design Optimization", *Journal of Mechanical Design*, c. 112, sy 2, ss. 223-229, Haz. 1990, doi: 10.1115/1.2912596.
[40] S. Gupta, K. Deep, H. Moayedi, L. K. Foong, ve A. Assad, "Sine cosine grey wolf optimizer to solve engineering design problems", *Engineering with Computers*, c. 37, sy 4, ss. 3123-3149, Eki. 2021, doi: 10.1007/s00366-020-00996-y.
[41] S.-J. Wu ve P.-T. Chow, "Genetic Algorithms for Nonlinear Mixed Discrete-Integer Optimization Problems via Meta-Genetic Parameter Optimization", *Engineering Optimization*, c. 24, sy 2, ss. 137-159, May. 1995, doi: 10.1080/03052159508941187.

BIOGRAPHY

Erdal Eker graduated in Mathematics from Van Yuzuncu Yil University, Turkey. He received his MSc in Applied Mathematics from Ataturk University, Turkey, and his PhD from Yuzuncu Yil University in Statistics. He is currently an instructor at Mus Alparslan University, working on Artificial learning, the applications of metaheuristic optimization, and statistic. He has many studies published in national and international scientific platforms.