New Simple Trigonometric Algorithms for Solving Optimization Problems

Baskar A[1*]

[1] Panimalar Institute of Technology, Poonamallee, Tamil Nadu 600123, India
[*] Corresponding author. E-mail: a.baaskar@gmail.com

Received: Dec. 11, 2021; Accepted: Feb 02, 2022

Heuristics play a key role in solving optimization problems with complex functions. They are popular as efficient heuristics are capable of providing results quickly with acceptable solution quality. Population-based heuristics are stochastic and hence, several iterations and trials are needed to achieve the expected accuracy and convergence to the global optimum. This article proposes two new, simple; population-based trigonometric algorithms, Sine (AB) and Cosine (AB). The algorithms are validated using forty well-known benchmark test functions available in the literature. The results are compared with a similar popular Sine Cosine Algorithm and the computational results show that the performance of Sine (AB) and Cosine (AB) are better than Sine Cosine Algorithm. Wilcoxon Signed-Rank and Friedman tests are carried out for statistical analyses. In addition to unconstrained functions, three real-world, constrained problems are solved to have a more intensive analysis of the proposed algorithms.

Keywords: Optimization, Population-based heuristic, Un-constrained optimization, Sine Cosine Algorithm, Trigonometric Algorithm

http://dx.doi.org/10.6180/jase.202212_25(6).0020

1. Introduction

Optimization algorithms are effectively used in solving real-world problems including the engineering domain. Different modelling and computing techniques are developed over the decades to meet the requirement of different types of optimization problems. Major categories of approaches are classical techniques, linear programming, nonlinear programming, geometric programming, dynamic programming, integer programming, stochastic programming, evolutionary algorithms, etc. The real optimization problem may be constrained or unconstrained; single objective or multi-objective; deterministic or stochastic.

The constrained problems may involve unknown Lagrange multipliers. The first application of the steepest descent method to solve unconstrained optimization problems was studied by Cauchy. Linear programming and dynamic programming were introduced in the 1940s and 1950s. The work by Kuhn and Tucker in 1951 on the necessary and sufficient conditions for the optimal solution and the contributions of Zoutendijk and Rosen in the early 1960s were important milestones in the development of non-linear programming.

In the 1960s, geometric programming was developed by Duffin, Zener, and Peterson. Some pioneering work in integer programming which is applied to most real-world problems was done by Gomory. Later; Dantzig, Charnes and Cooper developed stochastic programming techniques. The necessity to optimize more than one objective resulted in the development of multi-objective programming methods like Goal programming [1]. New classes of mathematical programming techniques that have come into prominence in recent times include simulated annealing, evolutionary algorithms including genetic algorithms and neural network methods.

Due to the complex nature of the optimization problems, exact methods may not be possible always. This makes way to heuristic and metaheuristic methods and is widely stud-

ied domains of Operational Research, Computer Science and Artificial Intelligence (Burke et al., 2013) [2]. Several stochastic metaheuristic algorithms have been proposed (Kumar et al., 2014) [3] over the years and are available in the literature for solving such complex problems. They comfortably replace the traditional methods in many cases.

Optimization algorithms can be broadly classified into local search and population-based methods. Local search methods have good search-ability (exploitation) whereas; the disadvantage is their focus on local search rather than global search (exploration). As a result, the probability of getting stuck in the local optimum is high for these algorithms (Abualigah and Diabat, 2021) [4].

Recently, population-based stochastic algorithms are gaining more popularity due to their simplicity and effectiveness. Mostly, they are problem independent and may require a few or no tuning parameters. Stochastic algorithms usually consider the optimization problems as black boxes (Droste et. al., 2006) [5] and hence, derivatives of functions are not required. The random solutions initially generated are evaluated for the objective (cost) function which is the basis for this optimization technique (Pal et al. 2019) [6]. Several trials are usually required to reach the global optimum or near the global optimum in many cases.

Another way of classifying the optimization metaheuristics can be (i) Evolutionary Algorithms (ii) Physics-Based and Maths-Based Algorithms (iii) Swarm-Based Algorithms and, (iv) Human-Based Algorithms which is presented in Fig. 1.

The JAYA algorithm proposed by Rao (2016) [7] is a simple yet powerful optimization algorithm that does not require any tuning parameter. It is widely used in solving real-world engineering problems also. JAYA is also a population-based metaheuristic that builds the initial population by generating two random numbers. This is one of the extensively studied algorithms in recent times (Zitar et al., 2021) [8].

The Sine Cosine Algorithm (SCA) proposed by Mirjalili (2016) [9] is also an efficient population-based algorithm. It has a time complexity of O(No_Of_Iterations*Population_Size*No_Of_Decision_Variables). It initially generates a pool of candidate solutions using sine and cosine functions and the 'best solution' is iteratively moved towards the optimum solution. Three random numbers are generated by the algorithm during the solution process. The performances of both JAYA and SCA were analysed by Kommadath et al. (2017) [10] and the results show that both algorithms are competitive.

A nature-inspired meta-heuristic optimization algorithm, called Whale Optimization Algorithm (WOA), mimics the social behaviour of humpback whales, having a complexity of O(No_Of_Iterations*Population_Size*O(fitness)) was proposed by Mirjalili and Lewis (2016) [11]. The bubble-net hunting strategy inspired algorithm was tested with 29 mathematical optimization problems and 6 structural design problems. The binary versions of the Whale Optimization Algorithm (bWOA-S and the bWOA-V; Hussien et al., 2020) [12] were tested with 22 benchmark functions and three engineering design problems. The results were compared with five well-known metaheuristic algorithms and statistically analyzed.

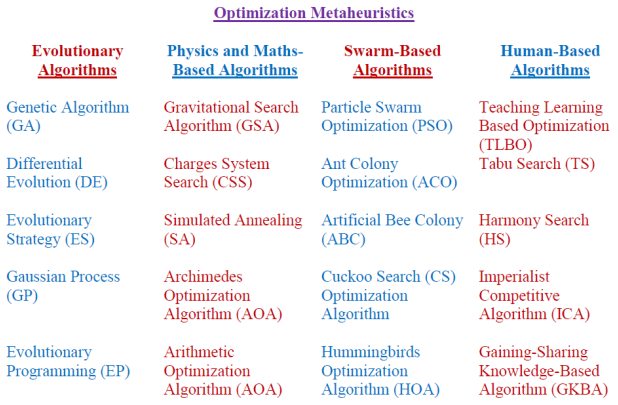


Fig. 1. One Classification of Optimization Metaheuristics with Examples

Another group of optimization methods is human-based which are influenced by human behaviour in communities. Imperialist Competitive Algorithm by Atashpaz-Gargari and Lucas (2007) [13] and Teaching–learning-based optimization algorithm of Rao et al. (2011) [14] are examples for this group.

The Arithmetic Optimization Algorithm (AOA) proposed by Abualigah et al. (2021) [15] utilizes the distribution behaviour of the four arithmetic operators in the optimization process. The time complexity is claimed to be O(Population_Size*(No_Of_Iterations*(No_Of_Decision_Variables+1)). The performances were compared with eleven other well-known algorithms and showed very promising results.

Hashim et al. (2021) [16] proposed a physics-inspired Archimedes Optimization (AOA) metaheuristic algorithm and the solutions obtained have outperformed a few well-known state-of-the-art and recently introduced metaheuristic algorithms. The AOA was analyzed using CEC'17 test suite and four engineering design problems. AOA was compared with a few popular algorithms; GA, PSO, L-SHADE, LSHADE-EpSin, WOA, A HHO, EO and SCA. It was shown that AOA outperformed other well-established

optimization methods. It was reported that SCA is a moderate performer. For the engineering problems, SCA performed better than GA, PSO, L-SHADE, LSHADE-EpSin and WOA. Agushaka and Ezugwu [17] proposed an advanced arithmetic optimization algorithm and tested the performance of the proposed nAOA with 20 classical, 10 composite test functions and three engineering design benchmarks. The algorithms used for comparison are: CP-SOGSA, GSA, PSO, BBO DE, ACO, SSA, SCA GWO, nAOA and AOA. It was concluded that nAOA outperformed most of the algorithms.

Inspired by the foraging process of hummingbirds, another "hummingbirds optimization algorithm (HOA)" was proposed by Zhuoran et al.[18] which was tested against CS, GSA, FA, GWO, MFO, CSA, SCA and SBO. Ten classic benchmark functions and two engineering design optimization problems were used for the comparison and the results showed that the HOA outperformed other algorithms. Recently, Zhao et al. [19] proposed an "Artificial hummingbird algorithm (AHA)" based on the flight skills and three foraging behaviour of hummingbirds. Both benchmark functions and constrained engineering design problems are used for the analysis.

Usually, algorithms are being developed in three ways; modifying the existing better-performing algorithms, developing hybrid algorithms and, crafting new novel algorithms. All three strategies are followed by researchers as "no single algorithm can solve all optimization problems" according to the 'No Free Lunch Theorem' (Wolpert and Macready, 1997) [20].

Any new algorithm is tested using a specific set of benchmark functions and compared against a few algorithms. Though this may not reveal the true power of the new algorithm, this can give a fair idea of the performance. The chain of comparisons grows resulting in specific groups of algorithms with defined properties.

This article, influenced by the SCA algorithm, significantly modifies the SCA and separately uses the sine and cosine functions for updating the positions.

2. Mathematical Representation

Optimization problems involve finding the best of an exponentially large set of solutions under the existing conditions. The general form of any optimization problem is [21]:

$$Minimize \quad f(x) \quad subject \quad to \quad x \in \Omega \tag{1}$$

where f, is the scalar-valued objective function, x is the decision variable and Ω is the feasible set or constraint set. If the objective function is to be maximized then,

$$Minimize - f(x) \quad subject \quad to \quad x \in \Omega \tag{2}$$

The feasible set in its functional form,

$$\Omega = \left\{ x \in \mathbb{R}^n | g_i(x) \leq 0, i = 1, ..., m; h_j(x) = 0, j = 1, ..., k \right\} \tag{3}$$

where, $g(x)$ and $h(x)$ represent inequality and equality constraints respectively.

The optimal or global solution x^* is a point in the feasible set that satisfies,

$$f(x^*) \leq f(x), \forall x \in \Omega \tag{4}$$

The optimization may be constrained or unconstrained. The unconstrained optimization corresponds to the case where $\Omega = \mathbb{R}^n$.

3. Sine Cosine Algorithm (SCA)

The SCA proposed by Mirjalili (2016) uses both the sine and cosine functions in its algorithm. The mathematical expression for SCA is:

$$\begin{aligned} X_{i+1} &= X_i + (r1^* (\sin(r2))^* abs\left(r3^* Best_i - X_i\right)), if r4 < 0.5 \\ X_{i+1} &= X_i + (r1^* (\cos(r2))^* abs\left(r3^* Best_i - X_i\right)) otherwise. \end{aligned} \tag{5}$$

where; X_{i+1} = Present values of decision variables in the population matrix
X_i = Previous values of decision variables in the population matrix
$Best_i$ = Decision variables' vector corresponding to the minimum cost function in the previous iteration
r1 = a – (t*a/T) … a = 3 or 2 in this work
r2 = (2*pi)*rand () … rand () = [0 1]
r3 = 2*rand (); r4 = rand ()
t = Current iteration number
T = Maximum no. of iterations.

The MATLAB codes provided by the original author consider a value of a = 2 and the original article gives another value of a = 3. Hence, both the values of '***a***' are considered in this article for the SCA algorithm. The success of SCA is due to its simple implementation, its good convergence speed, its reasonable execution time, and its ability to be easily hybridized with other optimization algorithms [22]. Motivated by these features, SCA was extensively studied by the author for its performance, speed and solution quality.

4. Methodology used and New Algorithms Developed

A general methodology for an optimization algorithm is generating an improved solution set from the previous results. A new solution set is constructed around the previous values by adding or subtracting random values within the permitted bounds in some logical pattern. That is,

$$New\ \ Solution\ \ Set = Old\ \ Solution\ \ Set \pm Fractions\ \ of\ \ Old\ \ Solution\ \ Values \tag{6}$$

To induce a fair amount of randomness, it has been decided to have three parts in the second term; tuning parameter, random values from -1 to +1 and an absolute term relating the previous solution set and the best solution obtained in the previous set. Due to the better performance of SCA, it has been decided to have the Sine or Cosine values as a part of the second part. The new algorithms are finalised as:

$$Sine(AB): \quad X_{i+1} = X_i + (r1^*(sine(r2))^*abs(r3^*Best_i - X_i)) \tag{7}$$

$$Cosine(AB): \quad X_{i+1} = X_i + (r1^*(cosine(r2))^*abs(r3^*Best_i - X_i)) \tag{8}$$

Where; X_{i+1} = Present values of decision variables in the population matrix
X_i = Previous values of decision variables in the population matrix
$Best_i$ = Decision variables' vector corresponding to the minimum cost function in the previous iteration
r1 = a – (t*a/T) … ***a*** = 1.5 or 1 in this work
r2 = (2*pi)*rand () … rand () = [0 1]
r3 = rand ()
t = Current iteration number
T = Maximum no. of iterations

Though they look similar to the SCA, significant differences are there. The first term of the RHS represents the previous results. The second term consists of three parts; '***r1***', sine (2*pi*rand)) or cosine (2*pi*rand)) and, absolute (rand*Besti – Xi)). The computing method for '***r1***' is the same but, the tuning parameter is different from SCA. New algorithms select a = 1.5 or a = 1 whereas, SCA goes with a = 3 or a = 2. The value of '***r2***' is the same for SCA and new algorithms. The value of '***r3***' is (2*rand) for SCA whereas, '***r3***' is just a 'random number [0, 1]' for the proposed algorithms. SCA has an additional random number, '***r4***' which is absent here. Another difference is that in SCA, both sine and cosine terms are integral parts whereas, in the proposed algorithms, only one trigonometric term is used.

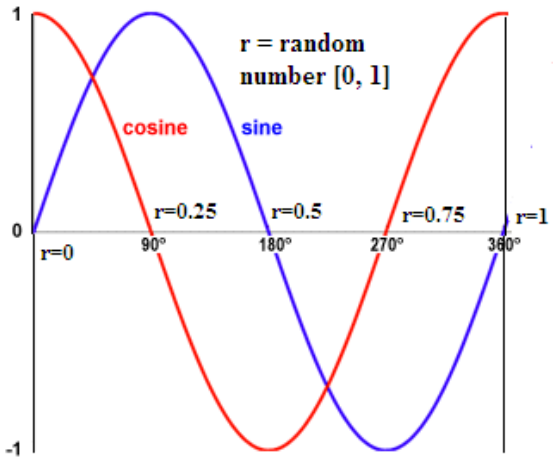


Fig. 2. Values of Sine (2*pi*rand)) and Cosine (2*pi*rand))

The first part '***r1***' reduces the tuning parameter '***a***' gradually to zero at the end of the maximum number of iterations. This narrows down the search space gradually irrespective of the net value of the other two parts. The refining parameter '***a***' value can be changed depending on the search span. For widely separated bounds, larger values can be used.

Similar to SCA, the variable '***r1***' decides the direction of movement of the solution. Its value equals '***a***' in the beginning and dynamically gets reduced as the iterations progress. At the end of maximum iterations, '***r1***' becomes zero.

The values of '***r2***' vary from 0 to 2π and hence, ***sine values*** vary from 0 to +1 to 0 to -1 to 0 and, ***cosine values*** vary from +1 to 0 to -1 to 0 to +1 (Fig. 2). That is, the sine curve is a cosine curve that has been shifted in the other direction by -90°. Hence, for the same search points, the solution values vary resulting in different solution sets for the new algorithms.

It should be '2π' so that the expression will always be $X_{i+1} = X_1 \pm (\boldsymbol{K})$ in both the algorithms. If it is changed to 'π', it will work fine for the Cosine (AB) only not for the Sine (AB) algorithm.

The third part varies from 'X_i' to $|Best_i - X_i|$. The value of '***r3*** $\in$ [0, 1]' does not influence the sign of '***K***' in the expression as only the absolute value of that particular term is taken.

Hence, instead of having both the sine and cosine terms in a single algorithm, only one term (either sine or cosine) is considered to get a good search for getting fairly improved solutions in the subsequent iterations.

For the proposed algorithms, the time complexity of generating the initial population is O(n). The complexity of performing the Sine (or) Cosine operation is O(T * n * d) and, the bound checking process has O(T * n). Hence, the net time complexity is: O(n) + O(T * n) + O(T * n * d). Considering the highest term,

Complexity: O(T * n * d) where; n – population size, d – dimension (number of variables) and, T – number of iterations.

The SCA though has the same complexity, has an additional process; checking whether the value of the additional random number is < 0.5 or not and accordingly, builds the new solution set. Hence, the execution time of SCA is slightly higher than the new algorithms.

In the computation process, values less than 1E-12 are considered zero.

Codes are generated in MATLAB 2012b and run in an i5 PC with 4 GB RAM. The codes provided by the author of SCA are used for the computations incorporating the new algorithms.

5. Benchmark Functions

Original SCA was tested using 19 benchmark functions and the results were analysed. In this work, 40 functions with varying dimensions and complexity levels are used in the validation process.

The 40 benchmark functions (Table 1) have 2 to 30 decision variables (dv). They include both uni-modal and multi-modal functions with varying difficulty levels. The search range and corresponding optimal (minima) are also listed.

The values of the recommended constants are also given in the search range column. Many functions are selected from the "Optimization" link available at [23].

6. Computational Results

To maintain the uniformity, the original codes used for the SCA algorithm was downloaded from the author's website [24] and the new algorithms are included by suitably modifying a few lines.

The population size is taken as 30 and uniformly 30 runs are conducted under the same computing conditions. The maximum number of iterations is fixed as 500 for each trial. Two sets of computations are carried out with two different values of the constant '***a***'; 3 and 2 for the algorithm SCA and, 1.5 and 1 for Sine (AB) and Cosine (AB).

Four performance metrics; '***best***', '***mean***', '***worst***' and '***standard deviations***' of the cost functions are estimated for every 30 trials and presented in Tables 2 to 5 for both SCA and Sine (AB) algorithms.

For a similar set of conditions, the results of Cosine (AB) are presented in Table 6 and Table 7.

The value of the tuning parameter, '***a***' can be changed depending on the search space to obtain a different set of results. Initially, it may be required to go for a few trials to fix its magnitude. In general, for a wider search space, we can go for a larger '***a***' value and a narrow search space, a smaller value of '***a***' may be adequate.

To check the converging pattern of the Sine (AB) algorithm, a few functions are separately run and the cost function values are noted in steps of 50 iterations (Table 8). The number of decision variables (dv) varies from 2 to 30.

The outputs are plotted (Iteration vs Function Values) and are presented in Fig. 3.

Fig. 3 shows that maximum convergence takes place within 100 (for the plotted functions) iterations and the convergence rate remains almost flat afterwards.

7. Analysis of Results

The results are analysed mathematically as well as statistically in this section.

The SCA algorithm was originally compared [9] with six other popular algorithms; PSO, GA, BA, FPA, FA and GSA. The performance of SCA was claimed to be superior to all followed by Particle Swarm Optimization (PSO) [25] and Firefly Algorithms (FA)[26]. The results were normalized in [0, 1] to compare the results of all benchmark test functions.

The performance measures are summarized in Table 9 regarding the optimal values. The deviations from the optimal values are computed for the '***best***' and '***mean***' values. The number of functions for which the optimal values are reported is counted and listed. In other cases where optimal results are not obtained, the number of cases in which the better results are reported is presented in the last column of Table 9.

Sine (AB) is the better performer for the '***best***' values with a sum of 69.85 when a = 1.5 and ahead of the other two in all three measures when a = 1.0. Cosine (AB) reports better results for the '***mean***' and '***SD***' values when a = 1.5.

When the number of optimal results and best results is considered, the Cosine (AB) performs better than the other two. Cosine (AB) reports optimal results in 13 of the 40 benchmark functions when a = 1.5 which comes down to 9 when a = 1.0. Sine (AB) yields optimal results in 11 and 8 functions respectively. SCA comes next with 2 and 8 cases. Similarly, Cosine (AB) is accountable for the best results in 25 and 24 test functions respectively. Sine (AB) reports the best results in 20 and 15 functions. Performance of SCA is better here with 9 and 14 best results.

Table 1. Benchmark Functions (Unconstrained Problems)

S.No.	Function	dv	Search Range	Optimal Value
1	Ackley	30	-32.768 to 32.768 (a=20, b=0.2, c=2π)	0
2	Beale	2	-4.5 to 4.5	0
3	Bohachevsky	2	-100 to 100	0
4	Booth	2	-10 to 10	0
5	Branin	2	X1$\in$[-5, 10]; X2$\in$[0, 15]	0.3979
6	Colville	4	-10 to 10	0
7	Cross in Tray	2	-10 to 10	-2.0626
8	Dixon Price	10	-10 to 10	0
9	Drop Wave	2	-5.12 to 5.12	-1
10	Easom	2	-100 to 100	-1
11	Egg Holder	2	-512 to 512	-959.6407
12	Goldstein-Price	2	-2 to 2	3
13	Griewank	4	-600 to 600	0
14	Hartmann3	3	0 to 1	-3.8627
15	Hartmann4	4	0 to 1	-3.1345
16	Hartmann6	6	0 to 1	-3.3223
17	Holder Table	2	-10 to 10	-19.2085
18	Langermann5	2	0 to 10 (m=5)	-4.1558
19	Levy	4	-10 to 10	0
20	Matyas	2	-10 to 10	0
21	Mccormick	2	X1$\in$[-1.5, 4]; X2$\in$[-3, 4]	-1.9132
22	Michalewicz2	2	0 to π (m=10)	-1.8013
23	Perm	2	-2 to 2 (β=0.5)	0
24	Powell	16	-4 to 5	0
25	Power Sum	4	0 to 4	0
26	Rastrigin	2	-5.12 to 5.12	0
27	Rosenbrock	30	-30 to 30	0
28	Rotated Hyper Ellipsoid	30	-65.536 to 65.536	0
29	Schaffer N4	2	-100 to 100	0.2926
30	Schaffer2	2	-100 to 100	0
31	Schwefel	2	-500 to 500	0
32	Shekel	4	0 to 10 (m=10)	-10.5364
33	Shubert	2	-5.12 to 5.12	-186.7309
34	Six Hump Camel	2	X1$\in$[-3, 3]; X2$\in$[-2, 2]	-1.0316
35	Sphere	30	-100 to 100	0
36	Styblinski-Tang	2	-5 to 5	-78.3323
37	Sum Spheres	30	-5.12 to 5.12	0
38	Three Hump Camel	2	-5 to 5	0
39	Trid6	6	-36 to 36 (d2)	-50
40	Zakarov	10	-5 to 10	0

Table 2. Results of a = 1.5 (3 for SCA)

	Sine(AB)				SCA			
Function	**Best**	**Mean**	**Worst**	**Std. Dev.**	**Best**	**Mean**	**Worst**	**Std. Dev.**
Ackley	4.59E-06	0.7652	4.6447	1.1656	6.3935	15.8354	20.0418	3.2337
Beale	7.97E-04	0.13	0.6141	0.1759	2.02E-04	0.0155	0.1035	0.0234
Bohachevsky	0.00E+00	0.2651	2.5667	0.6116	9.06E-06	3.6309	14.1636	3.8093
Booth	1.13E-06	0.0283	0.3651	0.0708	2.04E-08	1.33E-04	1.10E-03	2.84E-04
Branin	0.4011	0.6642	1.9981	0.3912	0.3989	1.0247	5.0401	0.9841
Colville	1.5732	8.4999	30.8927	7.1398	2.721	88.8632	495.9899	109.9822
Cross in Tray	-2.0626	-2.0618	-2.0587	0.0011	-2.0626	-2.0617	-2.06	8.50E-04
Dixon Price	0.6667	3.4276	47.5719	9.4716	19.2095	8.91E+03	5.40E+04	1.41E+04
Drop Wave	-1	-0.9983	-0.968	0.0062	-0.9999	-0.9492	-0.9335	0.0226
Easom	-0.9988	-0.6412	-0.0014	0.3169	-0.9929	-0.2901	-1.42E-14	0.3249
Egg Holder	-958.8964	-864.9411	-659.1166	73.8759	-959.64	-958.9899	-953.3996	1.2838
Goldstein-Price	3.0005	3.415	4.5413	0.475	3.0009	3.5161	5.4056	0.646
Griewank	0.00E+00	0.1777	1.9271	0.3806	0.0871	3.6013	9.8879	2.8459
Hartmann3	-3.8539	-3.3854	-2.074	0.5299	-3.8529	-3.1346	-1.6514	0.5485
Hartmann4	-3.0789	-2.5578	-2.0189	0.3189	-2.9599	-2.0443	-0.785	0.5895
Hartmann6	-2.9098	-2.4378	-1.9107	0.2719	-2.7224	-2.0672	-1.6159	0.3256
Holder Table	-19.1945	-17.8477	-15.3003	0.7367	-19.2069	-19.0828	-18.5268	0.1581
Langermann5	-4.1533	-3.7895	-2.4945	0.3016	-4.0697	-3.5514	-2.1772	0.3647
Levy	0.0305	0.1386	0.2922	0.0621	0.075	0.3021	0.7998	0.1934
Matyas	0.00E+00	6.99E-05	7.50E-04	1.95E-04	9.78E-09	0.0019	0.0076	0.0021

Table 3. Results of a = 1.5 (3 for SCA)

	Sine(AB)				SCA			
Function	**Best**	**Mean**	**Worst**	**Std. Dev.**	**Best**	**Mean**	**Worst**	**Std. Dev.**
Mccormick	-1.9129	-1.8771	-1.6618	0.0534	-1.9131	-1.9078	-1.8796	0.0072
Michalewicz2	-1.7924	-1.2781	-0.8881	0.3372	-1.7871	-1.1628	-0.6313	0.3201
Perm	5.40E-05	0.0401	0.2214	0.0611	3.42E-08	1.25E-04	2.10E-03	3.90E-04
Powell	4.03E-07	1.3987	33.5254	6.116	69.7979	1.10E+03	3.00E+03	7.52E+02
Power Sum	1.2318	22.5318	139.3736	24.8136	1.4846	60.7933	168.9145	57.3344
Rastrigin	0.00E+00	0.0086	0.1199	0.0251	9.69E-09	0.6407	1.7952	0.5551
Rosenbrock	28.7344	29.5151	35.6716	1.4519	169.8664	1.26E+03	7.10E+03	1.33E+03
Rotated Hyper Ellipsoid	7.35E-06	811.1024	2.03E+04	3.69E+03	1.84E+02	1.02E+05	3.18E+05	7.69E+04
Schaffer N4	0.2926	0.2932	0.2983	0.0013	0.2926	0.3022	0.329	0.0096
Schaffer2	0.00E+00	0.414	2.9484	0.8212	0.0218	3.867	6.977	2.0122
Schwefel	5.9678	141.7346	399.7884	93.385	0.5483	43.306	159.4908	49.0718
Shekel	-4.5978	-1.289	-0.5603	0.8249	-1.2614	-0.6625	-0.4089	0.1991
Shubert	-186.5724	-161.5731	-111.296	24.8176	-186.7127	-182.1532	-168.6305	4.7172
Six Hump Camel	-1.0316	-1.0275	-1.0016	0.0067	-1.0315	-1.0275	-1.0103	0.0053
Sphere	4.92E-07	87.4835	1.76E+03	325.8223	414.4065	1.49E+04	4.20E+04	1.26E+04
Styblinski-Tang	-78.3192	-74.5638	-62.5074	4.6046	-78.3301	-78.0278	-75.6937	0.5258
Sum Spheres	2.96E-06	16.4559	331.5034	61.218	12.9546	2.24E+03	8.43E+03	2.01E+03
Three Hump Camel	0.00E+00	7.43E-05	9.93E-04	2.43E-04	2.84E-10	0.0054	0.0317	0.0076
Trid6	-25.7206	-3.1232	21.816	8.4049	-5.7012	116.5381	298.9866	92.1191
Zakarov	0.00E+00	0.0677	0.9129	0.2152	0.0316	35.5466	109.5184	31.244
Sum	-1.25E+03	-14.8351	22205.4366	4341.8837	-387.6227	129246.5779	433059.7825	108016.2552

Table 4. Results of a = 1 (2 for SCA)

	Sine(AB)				SCA			
Function	**Best**	**Mean**	**Worst**	**Std. Dev.**	**Best**	**Mean**	**Worst**	**Std. Dev.**
Ackley	6.40E-03	0.6385	3.5067	1.0138	5.9181	15.6845	19.0183	3.1269
Beale	6.75E-04	0.1267	0.7086	0.1768	2.02E-04	0.017	0.0673	0.021
Bohachevsky	0.00E+00	0.1336	1.0914	0.2925	0.00E+00	1.4495	10	2.2877
Booth	9.35E-10	0.2056	2.0129	0.4938	1.63E-09	2.70E-03	3.23E-02	7.30E-03
Branin	0.3986	0.5267	1.1464	0.1734	0.3981	0.7932	2.1735	0.5887
Colville	1.5077	11.1598	28.3636	7.2923	2.0096	29.2946	152.2408	37.7964
Cross in Tray	-2.0626	-2.062	-2.059	8.73E-04	-2.0626	-2.0622	-2.0607	5.13E-04
Dixon Price	0.6722	2.665	45.1729	8.0552	0.7882	3.77E+03	1.95E+04	5.10E+03
Drop Wave	-1	-0.9948	-0.9362	0.014	-1	-0.9672	-0.9357	0.0279
Easom	-0.9926	-0.7634	-0.1891	0.232	-0.9945	-0.4907	-1.60E-12	0.3785
Egg Holder	-959.6259	-840.3078	-633.7672	86.3642	-959.6406	-941.7983	-841.6195	34.4996
Goldstein-Price	3.0001	4.8592	25.2895	5.5074	3.0028	3.4135	4.7635	0.4768
Griewank	9.40E-10	0.1679	0.7162	0.1943	4.10E-03	1.9243	5.987	1.4515
Hartmann3	-3.8512	-3.5538	-2.5644	0.3084	-3.8471	-3.2617	-2.0143	0.4956
Hartmann4	-3.0922	-2.8847	-2.4696	0.1701	-2.9648	-2.1684	-0.9679	0.4773
Hartmann6	-2.8879	-2.6361	-2.2441	0.1674	-2.8383	-2.3051	-1.6774	0.3144
Holder Table	-19.1649	-17.2712	-8.1915	2.0819	-19.203	-18.8231	-18.0197	0.3953
Langermann5	-4.1541	-3.995	-3.693	0.1481	-4.0514	-3.5291	-1.7597	0.5144
Levy	0.0697	0.155	0.3549	0.0556	0.0633	0.2384	0.8417	0.1822
Matyas	0.00E+00	2.07E-05	2.78E-04	5.52E-05	1.52E-09	1.50E-03	0.0165	3.10E-03

Table 5. Results of a = 1 (2 for SCA)

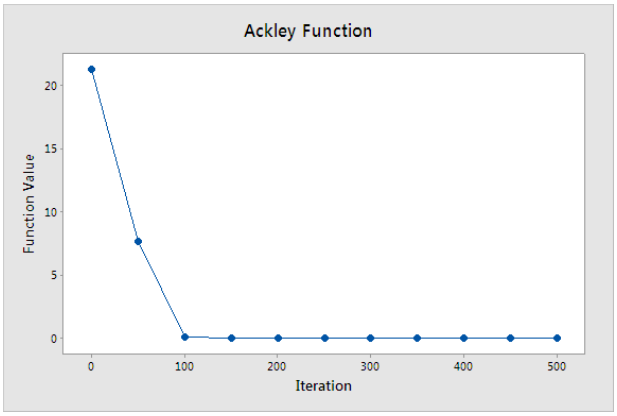
	Sine(AB)				SCA			
Function	**Best**	**Mean**	**Worst**	**Std. Dev.**	**Best**	**Mean**	**Worst**	**Std. Dev.**
Mccormick	-1.9129	-1.8622	-1.524	0.0806	-1.9132	-1.9079	-1.887	0.0075
Michalewicz2	-1.7957	-1.5323	-0.9869	0.2823	-1.8007	-1.262	-0.8945	0.3092
Perm	4.01E-05	0.0642	0.3541	0.095	2.39E-08	1.70E-03	3.26E-02	6.00E-03
Powell	8.29E-09	1.0157	18.9727	3.6265	67.1102	9.14E+02	2.71E+03	6.06E+02
Power Sum	0.0523	15.3864	82.2304	21.4451	0.3676	37.3002	153.4036	38.8761
Rastrigin	0.00E+00	0.018	0.2664	0.0554	3.39E-12	0.3299	1.3949	0.457
Rosenbrock	28.8656	33.0349	66.8422	9.1523	67.8771	1.15E+03	4.64E+03	9.68E+02
Rotated Hyper Ellipsoid	2.40E-03	468.4603	8.97E+03	1.65E+03	3.42E+02	1.05E+05	2.73E+05	7.19E+04
Schaffer N4	0.2926	0.2929	0.2954	6.69E-04	0.2926	0.2967	0.3082	0.0049
Schaffer2	8.54E-09	0.595	2.8588	1.0354	1.01E-04	3.0617	7.2183	2.4018
Schwefel	3.7099	190.1531	426.3673	90.7821	2.2649	46.6439	204.2811	51.8418
Shekel	-4.5318	-2.3737	-0.9625	1.0806	-1.7576	-0.8329	-0.4102	0.337
Shubert	-186.6471	-169.0174	-118.2284	21.6903	-186.7255	-181.2011	-136.979	10.189
Six Hump Camel	-1.0316	-1.0259	-0.9986	0.0091	-1.0315	-1.0288	-1.0169	0.004
Sphere	6.70E-03	41.8398	2.83E+02	78.3735	554.0422	1.27E+04	3.45E+04	1.03E+04
Styblinski-Tang	-78.2526	-74.9657	-63.4959	4.5905	-78.3215	-77.6299	-72.5599	1.2043
Sum Spheres	1.71E-04	27.3596	273.2224	71.7544	250.9367	2.55E+03	6.56E+03	1.47E+03
Three Hump Camel	0.00E+00	7.48E-05	6.93E-04	1.74E-04	8.51E-12	0.0021	0.0217	0.0046
Trid6	-29.4104	-2.9576	3.8614	8.0815	-24.6444	50.2349	222.4722	58.2703
Zakarov	3.77E-08	0.1047	0.7644	0.191	0.9431	24.3238	112.4982	20.3873
Sum	-1.26E+03	-329.2409	9393.6549	2078.8686	5.4747	125295.008	340330.0693	90663.1290

Table 6. Results of Cosine (AB) for a = 1.5

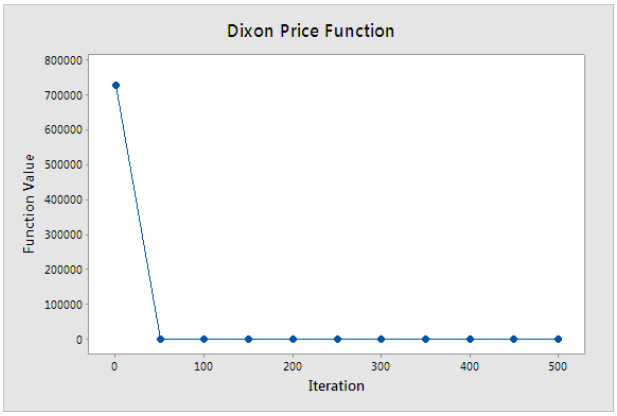
Function	Best	Mean	Function	Best	Mean
Ackley	3.43E-07	0.7883	Mccormick	-1.9126	-1.8643
Beale	5.87E-05	0.1944	Michalewicz2	-1.8006	-1.3348
Bohachevsky	0.00E+00	0.1495	Perm	5.11E-09	0.0221
Booth	8.37E-07	0.0807	Powell	4.78E-12	3.7347
Branin	0.398	0.683	Power Sum	0.6364	29.6556
Colville	1.6036	9.8129	Rastrigin	0.00E+00	0.0297
Cross in Tray	-2.0626	-2.0621	Rosenbrock	28.7591	31.3666
Dixon Price	0.6667	1.2973	Rotated Hyper Ellipsoid	6.72E-11	372.2093
Drop Wave	-1	-0.9947	Schaffer N4	0.2926	0.2928
Easom	-0.9975	-0.6957	Schaffer2	0.00E+00	0.3982
Egg Holder	-959.6388	-866.0607	Schwefel	5.5073	180.3528
Goldstein-Price 3	3.8761	Shekel	-4.2921	-1.4597	
Griewank	0.00E+00	0.1489	Shubert	-186.7087	-160.6208
Hartmann3	-3.8488	-3.4142	Six Hump Camel	-1.0316	-1.0246
Hartmann4	-3.1124	-2.5597	Sphere	0.00E+00	29.2228
Hartmann6	-2.8823	-2.3621	Styblinski-Tang	-78.3286	-74.5044
Holder Table	-19.1908	-17.904	Sum Spheres	5.29E-11	20.9477
Langermann5	-4.1468	-3.8975	Three Hump Camel	0.00E+00	1.63E-04
Levy	0.0095	0.1505	Trid6	-16.3581	-1.6994
Matyas	0.00E+00	1.51E-05	Zakarov	0.00E+00	0.066
			Sum	-1.25E+03	-456.9786

Table 7. Results of Cosine (AB) for a = 1.0

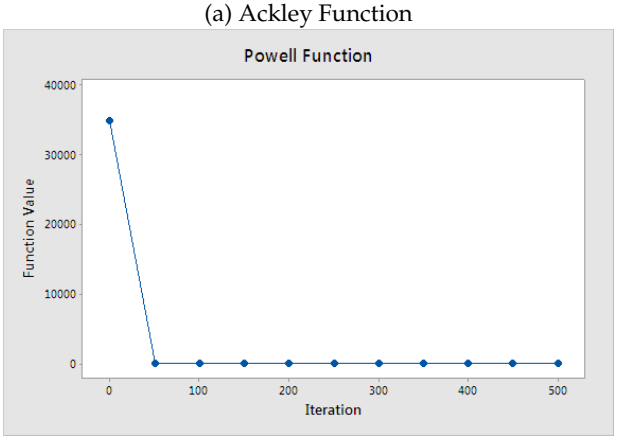
Function	Best	Mean	Function	Best	Mean
Ackley	0.0016	1.7678	Mccormick	-1.913	-1.8571
Beale	1.00E-03	0.1281	Michalewicz2	-1.7999	-1.3042
Bohachevsky	0.00E+00	0.1816	Perm	1.74E-07	1.10E-01
Booth	1.23E-05	1.59E-01	Powell	9.06E-10	1.03E+01
Branin	0.3983	0.4957	Power Sum	0.1432	11.4298
Colville	1.1162	12.593	Rastrigin	0.00E+00	0.0121
Cross in Tray	-2.0626	-2.0618	Rosenbrock	28.7605	3.59E+01
Dixon Price	0.667	6.04E+01	Rotated Hyper Ellipsoid	1.60E-04	7.66E+02
Drop Wave	-1	-0.9984	Schaffer N4	0.2926	0.2931
Easom	-0.9972	-0.7761	Schaffer2	2.92E-08	0.7165
Egg Holder	-958.832	-837.1387	Schwefel	4.4272	152.1594
Goldstein-Price 3	5.4636	Shekel	-4.0137	-2.4262	
Griewank	2.85E-10	0.1279	Shubert	-186.6968	-165.9744
Hartmann3	-3.8545	-3.6533	Six Hump Camel	-1.03E+00	-1.0136
Hartmann4	-3.1034	-2.866	Sphere	3.5755E-06	9.48E+01
Hartmann6	-2.8829	-2.5611	Styblinski-Tang	-78.3156	-73.256
Holder Table	-18.7508	-17.2991	Sum Spheres	5.06E-05	6.67E+00
Langermann5	-4.1507	-3.9954	Three Hump Camel	0.00E+00	2.86E-04
Levy	0.0336	0.1559	Trid6	-15.3483	-2.0012
Matyas	0.00E+00	4.98E-06	Zakarov	3.80E-11	0.165
			Sum	-1245.9116	40.5144



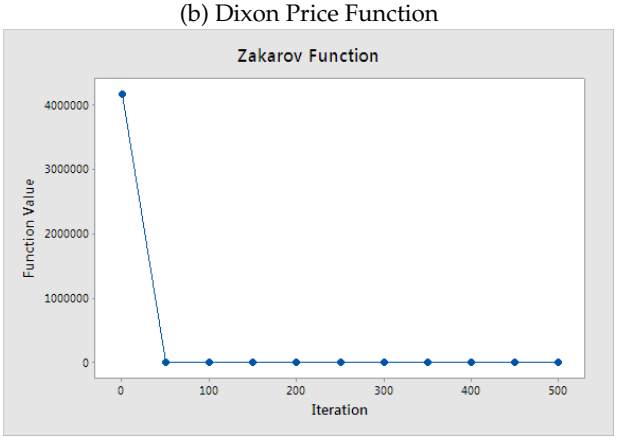
(a) Ackley Function



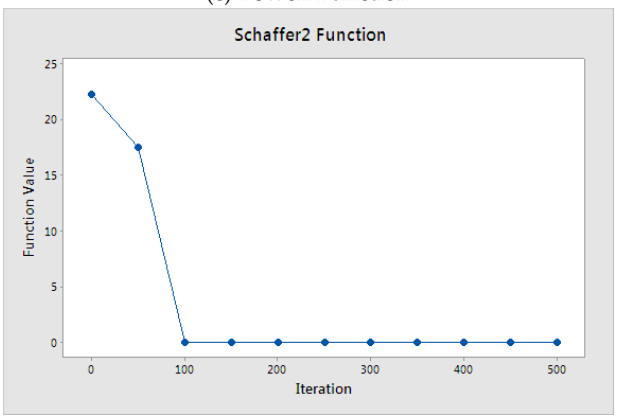
(b) Dixon Price Function



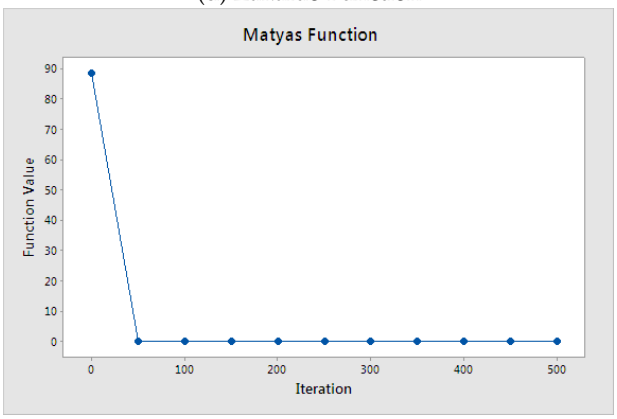
(c) Powell Function



(d) Zakarao Function



(e) Schaffer 2 Function



(f) Matyas Function

Fig. 3. The Function Values vs No. of Iterations Plots

Table 8. Convergence of Sine (AB) towards Optimal [a=1.5; 500 iterations; one trial]

Iteration	Ackley (dv = 30)	Dixon Price (dv = 10)	Powell (dv = 16)	Zakarov (dv = 10)	Schaffer2 (dv = 2)	Matyas (dv = 2)
0	21.245	726177.75227	34948.98305	4160420.18900	22.299	88.362
50	7.598	1.1107	0.95708	0.00031877	17.5605	4.7045e-06
100	0.050343	1.0003	0.0082954	1.6645e-09	0.0026645	7.6507e-09
150	0.0046266	0.8146	2.6944e-05	2.1348e-11	5.33e-06	0
200	0.00030951	0.68612	1.3308e-09	0	5.7365e-09	0
250	1.3602e-05	0.66671	1.0008e-11	0	1.4364e-11	0
300	4.6806e-06	0.66671	0	0	0	0
350	2.0601e-06	0.66668	0	0	0	0
400	8.1796e-07	0.66668	0	0	0	0
450	5.3806e-07	0.66667	0	0	0	0
500	5.1043e-07	0.66667	0	0	0	0

Table 9. Summary of Performance of Algorithms (40 Benchmark Functions)

Algorithm	**∑(Best-Optimal)**	**∑(Mean-Optimal)**	**∑SD**	**No. of Optimal Results***	**No. of Best Results**
		a = 1.5 (3 for SCA)			
SCA	936.4196107	130570.6202	108016.2552	2	9
Sine (AB)	**69.84666776**	1309.207244	4341.883738	11	20
Cosine (AB)	77.6032599	**867.0636781**	**1359.930483**	**13**	**25**
		a =1 (2 for SCA)			
SCA	1329.517003	126619.0503	90663.12901	5	14
Sine (AB)	**62.21388623**	**994.8013955**	2078.868571	8	15
Cosine (AB)	78.13072685	1364.556691	2843.630211	**9**	**24**

Note*: Less than 1E-12 is considered as zero

Both Sine (AB) and Cosine (AB) perform better than SCA for any performance metric considered.

7.1. Cost-Effectiveness

It is important to assess the performance of algorithms for optimisation problems in a way that balances solution quality and time. Cost-Effectiveness (CE) measures can be defined for both average-case and worst-case performance [27]. According to Beiranvand et al. [28], the performance measures fall into three categories: efficiency, reliability and quality of the algorithmic output. The number of function evaluations and running time come under efficiency, success rate and the number of global solutions fall under reliability and, computational accuracy is a measure of solution quality. Since the solution and optimal values are known, the combined efficiency could be defined as:

$$Accuracy, f_{acc} = \sum(f(\bar{x})) - f(x^*)) \tag{9}$$

To obtain the cost-effectiveness, this can be multiplied by the ratio, (average no. of iterations * execution time)/ (average no. of iterations of the competitor * execution time of the competitor). This ratio strikes a balance between the average number of iterations to get the output and the required running time. Mathematically,

$$Cost - Effectiveness, CE_{(1/2)} = f_{acc1} * (it_{avg1} * t_1)/(it_{avg2} * t_2) \tag{10}$$

CE essentially indicates the deviation from the optimal and hence, the smaller the better.

For '***best***' Values for a =1.5 (3 for SCA);

$Sine(AB)$:

$CE = 69.84666776^*(486.54167^*446.431909)/(500^*460.186158)$

$= 65.891157520.$

SCA :

$CE = 936.4196107^*(500^*460.186158)/(486.54167^*446.431909)$

$= 991.9709785.$

The complete computations for the SCA and Sine (AB) are presented in Table 10.

The average number of iterations for the Cosine (AB) is 484.362505 for a = 1.5 and 489.6841675 when a = 1.0. That is, in both cases, it is less than SCA and Sine (AB). The computation time does not vary from that of Sine (AB) and hence, the CE will be better than SCA and Sine (AB). The summary of results and CE confirm the better performance of Sine (AB) and Cosine (AB) over SCA.

Table 10. Cost-Effectiveness (CE) of SCA and Sine (AB) Algorithms

	a=1.5 (3 for SCA)				a=1 (2 for SCA)			
Algorithm	Accuracy	Avg. Iterations	Total Exe. Time, s	CE	Accuracy	Avg. Iterations	Total Exe. Time, s	CE
				Best Values				
Sine (AB)	69.84666776	486.54167	446.431909	65.89	62.21388623	490.918335	450.423649	60.76
SCA	936.4196107	500	460.186158	991.97	1329.517003	498.0433325	454.611595	1361.35
				Mean Values (computed simultaneously)				
Sine (AB)	1309.207244	486.54167	446.431909	1235.07	994.8013955	490.918335	450.423649	971.54
SCA	130570.6202	500	460.186158	138316.48	126619.0503	498.0433325	454.611595	129651.06

7.2. Statistical Analysis

The Wilcoxon signed-rank test is used for comparing paired data samples which is the nonparametric version of the paired Student t-test. The Friedman tests, on the other hand, are for comparing more than two data samples; the nonparametric version of the ANOVA and repeated measures ANOVA tests.

7.2.1. Wilcoxon Signed-Rank Test

The results of Wilcoxon signed-rank test is presented in Table 11 for the tuning parameter, a = 1.5 and 3 and in Table 12 for other values. The outputs, '***best***', '***mean***' and '***worst***', are compared with the optimal values individually for all three algorithms.

For a = 1.5 (3 for SCA), the P-Values are zero for all pairs indicating that the algorithms statistically differ from the optimal values. However, if the Wilcoxon Statistic and Estimated Median are considered, Cosine (AB) is ahead of SCA and Sine (AB) for the '***best***' and '***worst***' values whereas, Sine (AB) is marginally ahead of Cosine (AB) and well ahead of SCA for the '***mean***' values.

When the results for a = 1 (2 for SCA) are considered (Table 12), here also, the P-Values are zero for all pairs indicating that both the algorithms differ from the optimal values. However, if the Wilcoxon Statistic and Estimated Median are considered, Cosine (AB) is better than the other two for the '***best***' values and, Sine (AB) outperform others in the other two performance measures.

7.2.2. Friedman Test

The Friedman test is also conducted for assessing SCA (a = 3 and 2), Sine (AB) and Cosine (AB) [a = 1.5 and 1.0], separately for '***best***', '***mean***' and '***worst***' results (Table 13).

Friedman's test shows that the Sine (AB) and the Cosine (AB) algorithms are better than the SCA. For the '***best***' results, the Cosine (AB) is ahead of the other two in both cases of '***a***'. For the '***mean***' and '***worst***' values, Sine (AB) is the better performer followed by Cosine (AB). The computational results and statistical analyses show the competitive performances of Sine (AB) and Cosine (AB) when compared to the popular SCA.

SCA is one of the best population-based algorithms and newly proposed algorithms are very similar to SCA and hence, SCA is taken as the benchmark algorithm in this work.

The new algorithms have a limitation too. Fixing a suitable tuning parameter, '***a***' is a challenging task. The results may be different for the same benchmark function if the tuning parameter changes. Similarly, the performance of the algorithm does change for the changing tuning parameter. This is evident from the computational results. The results vary for a = 1.5 and a = 1.0 for the new algorithms. That is, the new algorithms inherit the same limitation of SCA also. The performance of the new algorithms are assessed for three benchmark real-world constrained engineering problems also; spring design, pressure vessel design and welded beam design problems.

8. Real-World Engineering problems

To further analyse the performance of the proposed algorithms, three real-world engineering problems are considered; design of Tension/Compression Spring (Sadollah et al., 2013) [29], Pressure Vessel (Kannan and Kramer, 1994) [30] and Welded Beam (Coello, 2000) [31].

In the spring problem, the weight of the spring has to be minimized with four constraints; outside diameter limits, surge frequency, minimum deflection, and shear stress. There are three independent variables; wire diameter (x1), coil diameter (x2), and number of active coils (x3) subject to the conditions (bounds) $0.05 \leq x1 \leq 2.0$, $0.25 \leq x2 \leq 1.3$, and $2.0 \leq x3 \leq 15.0$.

In the pressure vessel design, the cost is optimized with four design variables: shell thickness (x1), head thickness (x2), inner radius (x3), and cylinder length (x4). There are four constraints in the design and the bounds are, $0 \leq x1$, $x2 \leq 100$ and $10 \leq x3$, $x4 \leq 200$.

For minimizing the cost of welded beam design with six constraints (shear stress, bending stress in the beam, buckling load on the bar, end deflection of the beam, and

Table 11. Wilcoxon Signed Rank Test (a = 1.5, 3 for SCA)

	Best			**Mean**			**Worst**		
Pair	Wilcoxon Statistic	P-Value	Estimated Median	Wilcoxon Statistic	P-Value	Estimated Median	Wilcoxon Statistic	P-Value	Estimated Median
Optimal vs SCA	703.0	0.000	0.3000	820.0	0.000	7.920	820.0	0.000	10.74
Optimal vs Sine (AB)	435.0	0.000	0.01525	820.0	0.000	1.951	820.0	0.000	15.46
Optimal vs Cosine (AB)	302.0	0.000	0.006950	820.0	0.000	2.015	820.0	0.000	8.830

Table 12. Wilcoxon Signed Rank Rest (a = 1, 2 for SCA)

	Best			**Mean**			**Worst**		
Pair	Wilcoxon Statistic	P-Value	Estimated Median	Wilcoxon Statistic	P-Value	Estimated Median	Wilcoxon Statistic	P-Value	Estimated Median
Optimal vs SCA	561.0	0.000	0.3941	820.0	0.000	8.934	820.0	0.000	56.26
Optimal vs Sine (AB)	528.0	0.000	0.02181	820.0	0.000	1.747	820.0	0.000	11.43
Optimal vs Cosine (AB)	465.5	0.000	0.01680	820.0	0.000	4.114	820.0	0.000	18.11

Table 13. Friedman Test

		Best			**Mean**			**Worst**		
Algorithm	N	Est. Median	Sum of Ranks	Rank	Est. Median	Sum of Ranks	Rank	Est. Median	Sum of Ranks	Rank
a = 3 for SCA; a = 1.5 for Sine (AB) and Cosine (AB)										
SCA	40	0.00288	100.0	3	0.3386	99.5	3	0.7318	97.0	3
Sine (AB)	40	-0.00013	75.0	2	0.0314	68.5	1	0.3456	70.0	1
Cosine (AB)	40	0.0000	65.0	1	0.0266	72.0	2	0.3309	73.0	2
		Grand Median	0.00092		Grand Median	0.1322		Grand Median	0.4694	
a = 2 for SCA; a = 1.0 for Sine (AB) and Cosine (AB)										
SCA	40	0.00007	90.0	3	0.3831	96.0	3	0.9542	96.0	3
Sine (AB)	40	0.00000	80.0	2	0.0948	66.0	1	0.3523	67.0	1
Cosine (AB)	40	-0.00002	70.0	1	0.0975	78.0	2	0.3217	77.0	2
		Grand Median	0.00002		Grand Median	0.1918		Grand Median	0.5427	

side), four decision variables are considered. The bounds for the variables; weld thickness (x1), length of the attached part of the bar (x2), bar height (x3) and, bar thickness (x4) are $0.1 \leq$ x1, x4 ≤ 2.0 and $0.1 \leq$ x2, x3 ≤ 10.0.

For this analysis, a total of 5000 iterations are considered with a population size of 30. The refining parameter considered for SCA is, a = 2 and for the new algorithms, a = 1. The penalty method is used for solving these constrained optimization problems with a penalty parameter of 10^9. The results of 30 trials are presented in Table 14.

In the spring design problem, Sine (AB) outperforms other algorithms in all the performance measures. In the pressure vessel problem, Cosine (AB) performs well except for the '***Best***' value. For the welded beam design problem also, the performance of Cosine (AB) is better than the others except for the '***Standard Deviation***' metric. In the latter two cases, Sine (AB) comes next reporting the '***Best***' value for the pressure vessel problem and better '***Mean***' values for the welded beam design problem.

9. Conclusion and Future Work

This article proposes two new, simple but effective population-based trigonometric algorithms, Sine (AB) and Cosine (AB). The algorithms are validated using 40 benchmark functions and three real-world constrained engineering problems available in the literature. The number of decision variables varies from 2 to 30. Both Sine (AB) and Cosine (AB) algorithms are compared with the popular Sine Cosine Algorithm (SCA) and the results show that their performance is better than SCA. Wilcoxon signed-rank test and Friedman test are conducted to compare them statistically. Results show that the Cosine (AB) algorithm is marginally ahead of Sine (AB) in terms of solution quality. As it is widely agreed that there cannot be a single algorithm to solve all optimization problems, Sine (AB) or Cosine (AB) may not be the best ones but, they are better algorithms presented before the research community for their comments and further improvements. The convergence is rapid in the first 100 iterations and then remains flat. In a few cases, the algorithms do not reach the global minimum. Hence, some strategy has to be worked out to avoid the local minima. Future work is to improve the algorithms in this direction. They have to be evaluated for more constrained optimization problems and real-world engineering problems also.

Acknowledgement

The author sincerely acknowledges and thanks to the anonymous reviewer(s) and editors for their constructive comments in improving the technical contents and presentation of this paper.

Declaration of interests

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

[1] Website. https://nptel.ac.in/content/storage2/courses/105108127/pdf/Module_1.

[2] E. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, (2013) *"Hyper-heuristics: A survey of the state of the art"* **Journal of the Operational Research Society *64***(12): 1695–1724. DOI: 10.1057/jors.2013.71.

[3] V. Kumar, J. Chhabra, and D. Kumar, (2014) *"Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems"* **Journal of Computational Science *5***(2): 144–155. DOI: 10.1016/j.jocs.2013.12.001.

[4] L. Abualigah and A. Diabat, (2021) *"Advances in Sine Cosine Algorithm: A comprehensive survey"* **Artificial Intelligence Review *54***(4): 2567–2608. DOI: 10.1007/s10462-020-09909-3.

[5] S. Droste, T. Jansen, and I. Wegener, (2006) *"Upper and lower bounds for randomized search heuristics in black-box optimization"* **Theory of Computing Systems *39***(4): 525–544. DOI: 10.1007/s00224-004-1177-z.

[6] P. K. Pal, K. Deep, and A. K. Nagar. "Performance of Sine–Cosine Algorithm on Large-Scale Optimization Problems". In: *Decision science in action*. Springer, 2019, 139–154.

[7] R. Venkata Rao, (2016) *"Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems"* **International Journal of Industrial Engineering Computations *7***(1): 19–34. DOI: 10.5267/j.ijiec.2015.8.004.

Table 14. Performance in Real Engineering Problems

Problem	Result (Best Values), *xi*	Cost Function Value			
		Best	Mean	Worst	SD
Helical Spring					
SCA, 148577	0.0513922, 0.3495922, 11.7271316	0.012675	0.012708	0.012733	0.000016
Sine (AB), a=1, 149843	0.0517859, 0.3589993, 11.1595379	**0.012669**	**0.012688**	**0.012708**	**0.000009**
Cosine (AB), a=1, 149550	0.0516237, 0.3551298, 11.3896990	0.012672	0.012690	0.012713	0.000011
Pressure Vessel					
SCA, 149868	0.7791159, 0.3858501, 40.3578889, 199.5145234	5891.529527	5903.109052	5917.286692	5.751463
Sine (AB) 149669	0.7787019, 0.3850304, 40.3457153, 199.6552098	**5887.181959**	5894.397583	5905.200854	4.972476
Cosine (AB) 149873	0.7785406, 0.3849215, 40.3373646, 199.8167133	5887.795004	**5894.314754**	**5903.183230**	**3.126792**
Welded Beam					
SCA, 149813	0.2057433, 3.0244836, 9.0361975, 0.2057515	1.664216	1.666706	1.669744	0.001469
Sine (AB) 149862	0.2056668, 3.0239778, 9.0350225, 0.2058066	1.664252	**1.664922**	1.666274	0.000548
Cosine (AB) 149797	0.2056882, 3.0207320, 9.0392713, 0.2057421	**1.664078**	1.664958	**1.666057**	**0.000535**

[8] R. Zitar, M. Al-Betar, M. Awadallah, I. Doush, and K. Assaleh, (2021) *"An Intensive and Comprehensive Overview of JAYA Algorithm, its Versions and Applications"* **Archives of Computational Methods in Engineering**: DOI: 10.1007/s11831-021-09585-8.

[9] S. Mirjalili, (2016) *"SCA: A Sine Cosine Algorithm for solving optimization problems"* **Knowledge-Based Systems *96***: 120–133. DOI: 10.1016/j.knosys.2015.12.022.

[10] R. Kommadath, J. Dondeti, and P. Kotecha. "Benchmarking JAYA and sine cosine algorithm on real parameter bound constrained single objective optimization problems (CEC2016)". In: ***Part F127854***. cited By 6. 2017, 31–34. DOI: 10.1145/3059336.3059363.

[11] S. Mirjalili and A. Lewis, (2016) *"The Whale Optimization Algorithm"* **Advances in Engineering Software *95***: 51–67. DOI: 10.1016/j.advengsoft.2016.01.008.

[12] A. Hussien, A. Hassanien, E. Houssein, M. Amin, and A. Azar, (2020) *"New binary whale optimization algorithm for discrete optimization problems"* **Engineering Optimization *52***(6): 945–959. DOI: 10.1080/0305215X.2019.1624740.

[13] E. Atashpaz-Gargari and C. Lucas. "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition". In: cited By 1792. 2007, 4661–4667. DOI: 10.1109/CEC.2007.4425083.

[14] R. Rao, V. Savsani, and D. Vakharia, (2011) *"Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems"* **CAD Computer Aided Design *43***(3): 303–315. DOI: 10.1016/j.cad.2010.12.015.

[15] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. Gandomi, (2021) *"The Arithmetic Optimization Algorithm"* **Computer Methods in Applied Mechanics and Engineering *376***: DOI: 10.1016/j.cma.2020.113609.

[16] F. Hashim, K. Hussain, E. Houssein, M. Mabrouk, and W. Al-Atabany, (2021) *"Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems"* **Applied Intelligence *51***(3): 1531–1551. DOI: 10.1007/s10489-020-01893-z.

[17] J. Agushaka and A. Ezugwu, (2021) *"Advanced arithmetic optimization algorithm for solving mechanical engineering design problems"* **PLoS ONE *16***(8 August): DOI: 10.1371/journal.pone.0255703.

[18] Z. Zhang, C. Huang, H. Huang, S. Tang, and K. Dong, (2018) *"An optimization method: Hummingbirds optimization algorithm"* **Journal of Systems Engineering and Electronics *29***(2): 386–404. DOI: 10.21629/JSEE.2018.02.19.

[19] W. Zhao, L. Wang, and S. Mirjalili, (2022) *"Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications"* **Computer Methods in Applied Mechanics and Engineering *388***: DOI: 10.1016/j.cma.2021.114194.

[20] D. Wolpert and W. Macready, (1997) *"No free lunch theorems for optimization"* **IEEE Transactions on Evo-**

lutionary Computation *1*(1): 67–82. DOI: 10.1109/4235.585893.

[21] D. Bertsekas. *Nonlinear Programming*. 3rd ed. Athena Scientific, 2016.

[22] A. Gabis, Y. Meraihi, S. Mirjalili, and A. Ramdane-Cherif, (2021) *"A comprehensive survey of sine cosine algorithm: variants and applications"* **Artificial Intelligence Review *54***(7): 5469–5540. DOI: 10.1007/s10462-021-10026-y.

[23] *Virtual Library of Simulation Experiments: Test Functions and Datasets*. Website. https://www.sfu.ca/~ssurjano/optimization.html. 2013.

[24] Website. https://seyedalimirjalili.com/sca..

[25] R. Eberhart and J. Kennedy. "New optimizer using particle swarm theory". In: cited By 12748. 1995, 39–43.

[26] X.-S. Yang, (2010) *"Firefly algorithm, stochastic test functions and design optimization"* **International Journal of Bio-Inspired Computation *2***(2): 78–84. DOI: 10.1504/IJBIC.2010.032124.

[27] G. Farr, (2015) *"Cost-effectiveness of algorithms"* **Discrete Mathematics & Theoretical Computer Science *17***:

[28] V. Beiranvand, W. Hare, and Y. Lucet, (2017) *"Best practices for comparing optimization algorithms"* **Optimization and Engineering *18***(4): 815–848. DOI: 10.1007/s11081-017-9366-1.

[29] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, (2013) *"Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems"* **Applied Soft Computing Journal *13***(5): 2592–2612. DOI: 10.1016/j.asoc.2012.11.026.

[30] B. Kannan and S. Kramer, (1994) *"An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design"* **Journal of Mechanical Design, Transactions of the ASME *116***(2): 405–411. DOI: 10.1115/1.2919393.

[31] C. Coello Coello, (2000) *"Use of a self-adaptive penalty approach for engineering optimization problems"* **Computers in Industry *41***(2): 113–127. DOI: 10.1016/S0166-3615(99)00046-9.