Complex & Intelligent Systems (2021) 7:249–295
https://doi.org/10.1007/s40747-020-00189-6

ORIGINAL ARTICLE

A comparative study of social group optimization with a few recent optimization algorithms

Anima Naik[1] · **Suresh Chandra Satapathy**[2]

Received: 22 January 2020 / Accepted: 17 August 2020 / Published online: 23 September 2020

Abstract
From the past few decades, the popularity of meta-heuristic optimization algorithms is growing compared to deterministic search optimization algorithms in solving global optimization problems. This has led to the development of several optimization algorithms to solve complex optimization problems. But none of the algorithms can solve all optimization problems equally well. As a result, the researchers focus on either improving exiting meta-heuristic optimization algorithms or introducing new algorithms. The social group optimization (SGO) Algorithm is a meta-heuristic optimization algorithm that was proposed in the year 2016 for solving global optimization problems. In the literature, SGO is shown to perform well as compared to other optimization algorithms. This paper attempts to compare the performance of the SGO algorithm with other optimization algorithms proposed between 2017 and 2019. These algorithms are tested through several experiments, including multiple classical benchmark functions, CEC special session functions, and six classical engineering problems etc. Optimization results prove that the SGO algorithm is extremely competitive as compared to other algorithms.

Keywords Meta-heuristic · Benchmark functions · Optimization algorithms · Fitness evaluations

Abbreviations

Pop_size	Population size
Max_FEs	Maximum number of function evaluations
Fes	Function evaluations
RS test	Wilcoxon's rank-sum test

Introduction

The meta-heuristic optimization algorithm is a practical approach for solving global optimization problems. It is mainly based on simulating nature and artificial intelligence tools, invokes exploration and exploitation search procedures to diversify the search all over the search space and intensify the search in some promising areas. Flexibility and gradient-free approaches are the two main characteristics that make meta-heuristic strategies exceedingly popular for optimization researchers. From 1960s till date, several meta-heuristic optimization algorithms have been proposed. According to no-free-lunch (NFL) [1] theorem for optimization, none of the algorithms could solve all classes of optimization problems. This motivated many researchers to design new algorithms or modify/hybridize existing algorithms to solve different problems or provide competitive results, as compared to the current algorithms.

Meta-heuristic algorithms can be classified into mainly four categories: (a) evolutionary-based algorithm, (b) swarm intelligence-based algorithm, (c) human-based algorithm, and (d) physics and chemistry-based algorithm. Evolutionary algorithms mimic concepts of evolution in nature. The genetic algorithm (GA) [2] is the best example of an evolutionary algorithm that simulates the concepts of Darwinian theory of evolution. After that several other evolutionary algorithms have been proposed such as evolutionary strategy (ES) [3], and evolutionary programming (EP) [4], gene expression programming (GEP) [5, 6], genetic programming (GP) [7], covariance matrix adaptation evolution strategy

✉ Anima Naik
animanaik@klh.edu.in

Suresh Chandra Satapathy
suresh.satapathyfcs@kiit.ac.in

[1] Department of CSE, KL University, Hyderabad, Telangana, India

[2] School of Computer Engineering, KIIT Deemed To Be University, Bhubaneswar, Odisha, India

CMA-ES) [8], differential evolution (DE) [9], biogeography-based optimization (BBO) algorithm [10].

Swarm intelligence algorithms mimic the intelligence of swarms. Each swarm consists of a group of creatures. So, these algorithms originate from the collective behaviour of a group of creatures in the swarm. Many swarm intelligence algorithms are seen in the literature. These are particle swarm optimization (PSO) [11] inspired by bird flocking, ant colony optimization (ACO) [12] inspired by Ants behaviour while collecting food, artificial bee colony (ABC) [13] mimicked by Honey bee for collecting nectar, etc. Additionally, there are many more algorithms such as bacterial foraging(BF) [14], bat algorithm (BA) [15], firefly algorithm (FFA) [16], krill herb (KB) [17], cuckoo search (CS) [18], monkey search (MS) [19], bee colony optimization (BCO) [20], cat swarm [21], wolf search (WS) [22], ant lion optimizer (ALO) [23], grey wolf optimization (GWO) [24], whale-optimization algorithm (WOA) [25], crow search algorithm (CSA) [26], Salp swarm algorithm (SSA) [27], grasshopper optimization algorithm (GOA) [28], butterfly optimization algorithm (BOA) [29], squirrel search algorithm (SSA) [30], Harris Hawks optimization (HHO) [31].

Human based algorithms are mainly inspired by behaviors of human. Some of the popular algorithms are teaching–learning-based optimization (TLBO) [32], harmony search (HS) [33], Tabu (Taboo) search (TS) [34–36], group search optimizer (GSO) [37, 38], imperialist competitive algorithm (ICA) [39], league championship algorithm (LCA) [40], firework algorithm [41], colliding bodies optimization (CBO) [42], interior search algorithm (ISA) [43], mine blast algorithm (MBA) [44], soccer league competition (SLC) algorithm [45], seeker optimization algorithm (SOA) [46], social-based algorithm (SBA) [47], exchange market algorithm (EMA) [48], and group counselling optimization (GCO) algorithm [49, 50], social emotional optimization (SEO) [51], ideology algorithm (IA) [52], social learning optimization (SLO) [53], social group optimization (SGO) [54, 55], election algorithm (EA) [56], cultural evolution algorithm (CEA) [57], cohort intelligence (CI) [58], anarchic society optimization (ASO) [59], volleyball premier league algorithm (VPL) [60], socio evolution and learning optimization algorithm (SELO) [61].

Physical and chemical based algorithms are inspired by physical rules and chemical reactions of the universe. Some popular algorithms are simulated annealing (SA) [62], gravitational local search (GLSA) [63], big-bang big-crunch (BBBC) [64], gravitational search algorithm (GSA) [65], charged system search (CSS) [66], central force optimization (CFO) [67], artificial chemical reaction optimization algorithm (ACROA) [68], black hole (BH) algorithm [69], ray optimization (RO) [70] algorithm, small-world optimization algorithm (SWOA) [71], galaxy-based-search algorithm (GbSA) [72], curved space optimization (CSO) [73], water cycle algorithm(WCA) [74]. Spiral optimization (SO) [75], river formation dynamics (RFD) [76], sine cosine algorithm (SCA) [77], multi verse optimizer (MVO) [78], lightning attachment procedure optimization (LAPO) [79], golden ratio optimization method (GROM) [80].

Meta-heuristic algorithms are extensively recognized as effective approaches for solving large-scale optimization problems (LOPs). These algorithms provide effective tools with essential applications in business, engineering, economics, and science. Recently, many researchers of meta-heuristic algorithm have paid their attention to the solution of large-scale optimization problems [81]. However, the standard metaheuristic algorithms for solving LOPs suffer from a main deficiency which is the curse of dimensionality, i.e., the performance of algorithms deteriorates when dimension of problems increases. There are two major reasons for the performance deterioration of these algorithms: Firstly, increasing size of the problem dimension increases its landscape complexity and characteristic alteration. Secondly, the search space exponentially increases with the problem size; so, an optimization algorithm must be able to explore the entire search space efficiently; which is not a trivial task. It is motivating to consider these reasons and difficulties while proposing new approaches of tackling LOPs [82]. The existing algorithms for LOPs can be mainly classified into the following two categories [83]: (1) the cooperative coevolution methods for LOPs, (2) the methods with learning strategies for LOPs. Since several real-world applications are considered as optimization of a large number of variables, various meta-heuristic algorithms have been proposed to handle the large-scale optimization problems [84–89].

In the real world, it is common to face an optimization problem with more than three objectives. Such problems are called many-objective optimization problems (MaOPs) that pose great challenges to the area of evolutionary computation. The failure of conventional Pareto based multi-objective evolutionary algorithms in dealing with MaOPs motivates various new approaches. Deb et al. [90] suggest a reference-point-based many-objective evolutionary algorithm following NSGA-II framework that emphasizes population members that are nondominated, yet close to a set of supplied reference points. This approach is applied to several many-objective test problems and gets satisfactory results to all problems [90]. Lin et al. propose a balanceable fitness estimation method and a novel velocity update equation to compose a novel multi-objective particle swarm optimization algorithm to address the (MaOPs) [91]. Achieving a balance between convergence and diversity in many-objective optimization is a great challenge. Liu et al. suggest an evolutionary algorithm based on a region search strategy that enhances the diversity of the population without losing convergence [92]. A hybrid evolutionary algorithm based on knee points and reference vector adaptation strate-

gies (KnRVEA) is proposed to improve the convergence of solution where a novel knee adaptation strategy is introduced to adjust the distribution of knee points [93]. A new differential evolution algorithm (NSDE-R) capable of efficiently solving many-objective optimization problems, where the algorithms make use of reference points evenly distributed through the objective function space to preserve diversity and aid in multi-criteria-decision making was thus proposed [94]. Generally, the methods proposed for solving MaOPs can be roughly classified into three categories [95]. These are (1) multi/many-objective optimization algorithms based on dominance relationships. (2) multi/many-objective optimization algorithms based on decomposition strategy and (3) indicator-based evolutionary algorithms. The application of many-objective optimization can be demonstrated in stormwater management project selection, encouraging decision-maker buy-in [96]. Additionally, the application can also be seen in mixed-model disassembly line balancing along with multi-robotic workstation [97].

Many real-world optimization problems are challenging because the evaluation of solutions is computationally expensive. For those expensive problems, there are three kinds of models utilized in the meta-heuristic algorithms, i.e., the global model, the local model, and the surrogate ensembles [98]. Surrogate-assisted evolutionary algorithms are promising approaches to tackle this kind of problems. They use efficient computational models, known as surrogates, for approximating the fitness function in evolutionary algorithms. These are found successful applications not only in solving computationally expensive single or multi-objective optimization problems but also in addressing dynamic optimization problems, constrained optimization problems, and multi-modal optimization problems. Surrogate models have shown to be effective in assisting meta-heuristic algorithms for solving computationally expensive complex optimization problems. Examples of some surrogate-assisted optimization algorithms for expensive optimization problems can be found in [99–103].

In the year 2017–2019, some of the popular meta-heuristic algorithms were proposed. Those are Salp swarm algorithm (SSA), grasshopper optimization algorithm (GOA), lightning attachment procedure optimization (LAPO), golden ratio optimization method (GROM), butterfly optimization algorithm (BOA), squirrel search optimization algorithm (SSOA), Harris Hawks optimization (HHO), volleyball premier league algorithm (VPL), socio evolution and learning optimization algorithm (SELO). SGO algorithm was proposed in the year of 2016 by Satapathy et.al. The SGO algorithm is based on the social behavior of humans for solving a global optimization problem. Applications of the SGO algorithm are discussed in papers [104–109]. In this work, the authors plan to have an exhaustive comparative analysis with SGO, their own proposed algorithm, and algorithms that were developed from 2017 to 2019. The GOA algorithm mimics the behavior of grasshopper swarms and their social interaction. Applications of the GOA algorithm are elaborated in papers [110–115]. The SSA algorithm is inspired by the swarming behavior of salps when navigating and foraging in oceans. In [116–121], applications of SSA are highlighted. The LAPO algorithm is based on the concepts of the lightning attachment procedure. The application of LAPO is seen in paper [122–125]. The GROM algorithm is inspired by the golden ratio of plant and animal growth. The BOA algorithm mimics the food search and mating behavior of butterflies. The SSOA algorithm imitates dynamic foraging behavior of southern flying squirrels and their efficient way of locomotion. The HHO algorithm is based on cooperative behavior and the chasing style of Harris' hawks in nature. The VPL algorithm is inspired by competition and interaction among volleyball teams during a season. It also mimics the coaching process during a volleyball match. The SELO algorithm is inspired by the social learning behavior of humans organized as families in a societal setup.

In this paper, we have compared the performance of those nine algorithms developed between 2017 and 2019, to SGO, which exhibit similar characteristics. These algorithms are tested through several experiments using many classical benchmark functions, CEC special session functions, and six classical engineering design problems. The results of the experiments are tabulated, and inferences are drawn in conclusion.

The remaining paper is organized as follows: In "Preliminaries of SGO, SSA, GOA, LAPO, GROM, BOA, SSOA, VPL, HHO, and SELO", all algorithms are briefly summarized. In "Simulation and experimental results", simulation and experimental results are discussed, and the paper concludes with "Conclusion".

Preliminaries of SGO, SSA, GOA, LAPO, GROM, BOA, SSOA, VPL, HHO, and SELO

Social group optimization (SGO) algorithm

The SGO algorithm is based on the social behavior of human to solve complex problems. Each person represents a candidate solution empowered with some information (i.e., traits) and has an ability to solve a problem. The human traits represent the dimension of the person, which in turn represents the number of design variables of the problem. This optimization algorithm goes through two-phase: improving phase and acquiring phase. In the improving phase, an individual's knowledge (solution) level is improved based on the best individual influence. In the acquiring phase, the individual's knowledge (solution) level is improved by mutual interaction between individuals and the person who has the highest

knowledge level, as well as the ability to solve the problem under consideration. However, for a detailed description of SGO, anyone's paper can be referred to [54, 55]. Algorithm 1 details the flow of SGO.

Algorithm: SGO Algorithm

Start

Assume the N person P_i (i = 1,2,……,N) in D-dimensional search space,
Randomly distribute the entire persons in the group throughout the search space during initialization process.
Compute fitness value f_i for each person P_i based on the problem under concern

Step 1: Find the best person 'gbest' in the group
[minvalue, index]=min$\{f_i,\ i = 1,2,3,\ldots\ldots\ldots,N\}$
gbest=P(index,:)

for solving the minimization problem

Step 2: Initiate improving phase to update the knowledge of persons with the help of 'gbest'

Step 3: Initiate acquiring phase to further update knowledge of a person by randomly choosing a person from the group and following the 'gbest'.

Step 4: if
all persons have approximately similar to fitness values or reach termination criterion
then
terminate the search and display the optimized result for the chosen problem
else
goto step 2.
endif

Stop

SSA, GOA, LAPO, GROM, BOA, SSOA, VPL, HHO and SELO algorithms

Preliminaries of the above-listed algorithms can be found in the literature. The SSA can be referred to in [27]; the basic of GOA is in [28], BOA is well described in [29]. The basic of SSOA is in [30]. The basic of HHO is in [31], and basic of VPL is in [60]. SELO, LAPO, and GROM can be followed in [61, 79, 80], respectively.

Simulation and experimental results

We have carried out an extensive comparison of SGO with the other nine algorithms. Individually, six experiments have been conducted batch-wise, on selecting a few algorithms out of nine algorithms; they are compared with SGO. Finally, in the last experiment, an overall comparison analysis is conducted with all nine algorithms with SGO. In the first experiment, the LAPO and GROM algorithms are compared with the SGO algorithm by considering twenty-nine benchmark functions in a combination of unimodal, multimodal, fixed dimensional and composite benchmark functions. In the second experiment, BOA is compared with the SGO algorithm using thirty classical benchmark functions. In the third experiment, SSOA is compared with the SGO algorithm using twenty-six classical benchmark functions, and seven functions are taken from the CEC 2014 special session. In the fourth experiment, VPL is compared with the SGO algorithm using twenty-three classical benchmark functions. In the fifth experiment, SELO is compared with the SGO algorithm using fifty classical benchmark functions. In the sixth experiment, HHO, SSA, and GOA algorithms are compared with the SGO algorithm using twenty-nine classical benchmark functions. The selections of the batch of algorithms are made based on the benchmark functions experiments which have been made by reference papers and the availability of results. In the seventh experiment, all algorithms are compared with each other by considering six classical engineering problems.

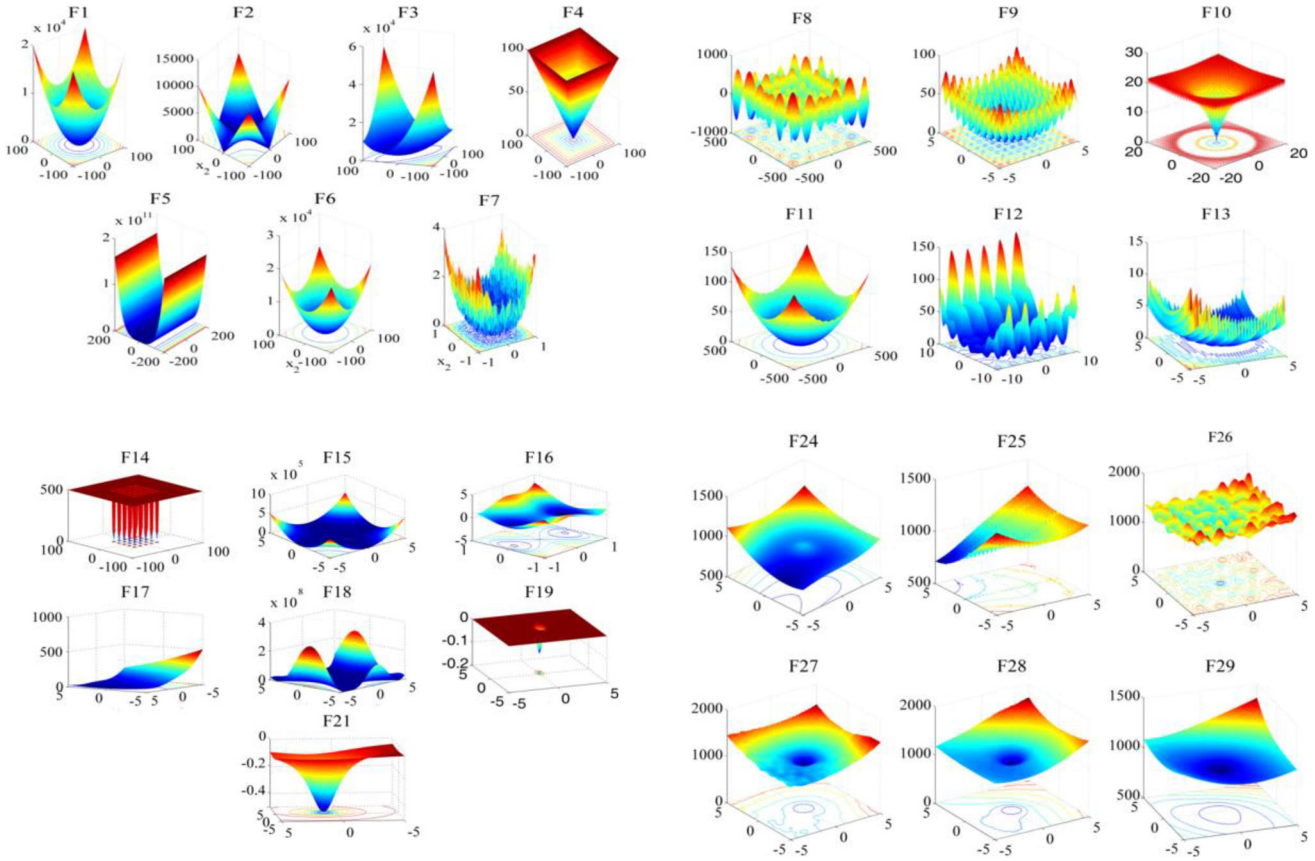

Fig. 1 Figure of benchmark functions

To compare the performance, the SGO algorithm is implemented by us and, the results of all other algorithms are taken from their respective papers (Fig. 1).

"For comparing the speed of the algorithms, the first thing we require is fair time measurement. The number of iterations or generations cannot be accepted as a time measure since the algorithms perform the different amount of works in their inner loops, and they have different population sizes (pop_sizes). Hence, we choose the number of fitness function evaluations (FEs) as a measure of computation time instead of generations or iterations" [54]. Since meta-heuristic algorithms are stochastic in nature, the results of two successive runs usually do not match. Hence, we have taken different independent runs (with different seeds of random number generator) of each algorithm and find out the best function value, mean function value, and standard deviation and put in tables in each experiment. For comparing the performance of algorithms, different tests have been conducted in experiments.

Experiment 1

In this experiment, LAPO and GROM algorithms are compared with the SGO algorithm. For comparison of the performance of algorithms, twenty-nine classical benchmark functions are considered. Out of which seven are unimodal benchmark-functions. The unimodal functions (F1–F7) are suitable for benchmarking the exploitation of algorithms since they have only one global optimum. Six are multimodal benchmark-functions and, ten are fixed-dimensional multimodal benchmark-functions. Each multimodal function from F8–F23 has massive numbers of local optima. These functions are considered to examine the exploration capability of algorithms. There are six composite benchmark functions. The composite benchmark-functions (F24–F29) are considered from CEC 2005 special session [126]. These benchmark functions are kept for judging the capability of the algorithm for the proper balance between exploration and exploitation search to avoid local optima and are described in Appendix A with illustrations in Fig. 2.

In our experiment test functions are solved for two cases, low dimensional and high dimensional. SGO algorithm is implemented in MATLAB 2016a. Experiments are conducted on an Intel Core i5, 8 GB RAM, and Windows 10 environment. For the LAPO, results are taken from [79], and for the GROM algorithm, the results are taken from [80].

For low dimensional cases, the common control parameter such as pop_size is set to 40, maximum iteration is 500, and

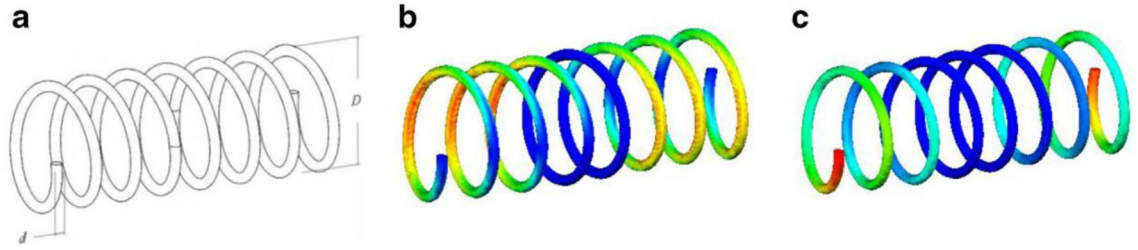

Fig. 2 **a** Schematic of the spring; **b** stress distribution evaluated at the optimum design; and **c** displacement distribution evaluated at the optimum design

Max_FEs is set to 40,000. The other specific parameters for each algorithm are given below.

- SGO setting: For SGO, there is only one parameter C called a self-introspection factor. The value of C is empirically set to 0.2.
- LAPO settings: There is no such specific parameter to set value.
- GROM settings: There is no such specific parameter to set value.

For each benchmark function, algorithms are run 30 times with different randomly generated populations. Statistical results in terms of best value, mean value, and corresponding standard deviation are reported in tables. Table 1 is for unimodal benchmark-functions. Table 2 is for multimodal benchmark-functions. Table 3 is for fixed dimensional multimodal benchmark-functions and, Table 4 is for the composite benchmark.

For the high dimensional case, the 200-dimensional version of unimodal, multimodal functions are solved in two cases. In case 1, pop_size is taken as 200, and the maximum iteration is 2000, and the results are given in Table 5. In case 2, pop_size is taken as 40, and the maximum iteration is 500, and the results are given in Table 6. The idea is to see how the algorithms behave for high dimensions in a large population with more iteration, and relatively small populations with less iteration. For every benchmark function, the best results have been put in bold letters in the result table.

To obtain statistically sound conclusions, Wilcoxon's rank-sum (WRS) test at a 0.05 significance level is conducted on experimental results, and the last three rows of each respective table summarize experimental results.

Discussion

The unimodal functions have only one global optimum. These functions allow evaluating the exploitation capability of the investigated meta-heuristic algorithms. As seen in Table 1, SGO has gained the best performance, and it has reached to first rank among other algorithms. SGO has shown excellent performance in exploitation capability and convergence characteristic. It has also successfully overcome to solve all the problems within this category except F5. It is clear from results that SGO achieves success in finding global optimum on F1, F2, F3, and F4 within 40,000 max_FEs. For F1, F2, F3, F4, F6, F7, the performance of SGO is better than LAPO and GROM algorithm, whereas, for F5, the performance of LAPO and GROM algorithm is far better than SGO. From Table 1, we find that out of seven unimodal test functions according to the WRS test, SGO performs superior to LAPO and GROM in six test functions and worse in one test function.

The multimodal functions test functions F8–F13 are beneficial, while the exploration capability of the optimization algorithm is considered. Form Table 2, results show that the SGO algorithm is eligible for solving problems with challenging search space. In this case, SGO has demonstrated excellent performance in comparison, and it has reached to first rank among algorithms. Table 2 shows that SGO has consistently performed better than other algorithms. SGO has an excellent performance in exploration, and it successfully overcomes to solve all the problems within this category. It is clear from table results that both SGO and GROM achieve success in finding global optimum on F9 and F11, and for F10, find equivalent results. From Table 2, we get that out of six multimodal test functions according to the WRS test, SGO performs superior to LAPO in all six test functions and superior to GROM in three test functions and equivalent in three test functions.

The fixed-dimensional multimodal functions are designed to have many local optimal where computation complexity increases drastically with the increase of the problem size. The results reported from Table 3 for functions F14-F23 indicate that SGO has an excellent exploration capability except for the shekel family (F21, F22, F23) that SGO has. It is clear from tabular results that the SGO algorithm achieves success in finding a global optimum on F14–F20. In contrast, GROM achieves success in F16–F19, F21–F23, and LAPO achieves success in F4, F16, F18, and F19 in finding a globally optimal solution. GROM achieves success in finding an optimum solution on the shekel family only. The

Table 1 Unimodal benchmark function on 30 repetitions

Functions	LAPO		GROM		SGO
	Results	WRS Test	Results	WRS Test	results
F1					
Best	1.3406e−15	–	**0**	−	**0**
Mean	2.0664e−13		3.1350e−247		**0**
Std	5.5098e−13		6.5961e−251		**0**
F2					
Best	4.2412e−9	−	2.8836e−53	−	**0**
Mean	2.2547e−8		1.367e−52		**0**
Std	1.7473e−8		9.173e−53		**0**
F3					
Best	4.1270e−7	−	**0**	−	**0**
Mean	1.1385e−5		3.398e−87		**0**
Std	3.6624e−5		9.6154e−87		**0**
F4					
Best	2.7951e−7	−	7.155e−145	−	**0**
Mean	4.3915e−7		1.118e−131		**0**
Std	1.3825e−7		3.0352e−132		**0**
F5					
Best	19.5667	+	**18.46243**	+	22.4419
Mean	22.7427		**19.12**		24.1058
Std	0.6846		**0.6160**		0.6143
F6					
Best	1.3619e−6	−	1.594e−7	−	**1.7698e−12**
Mean	1.1151e−5		3.9096e−7		**4.0337e−08**
Std	1.0200e−5		3.573e−7		**9.9545e−08**
F7					
Best	1.3323e−4	−	2.677e−5	−	**4.5252e−06**
Mean	7.1418e−4		0.000131		**3.0393e−05**
Std	4.3695e−4		4.7138e−5		**1.7084e−05**
Total +		01		01	
Total −		06		06	
Total ≈		00		00	

"−", "+" and "≈" denote that performance of LAPO and GROM is worse, better and similar to SGO, respectively

results for LAPO on the shekel family are better than the SGO algorithm. From Table 3, as per WRS test, we find that out of ten fixed dimensional multimodal test functions, SGO performs superior to LAPO in two test functions, worse in three test functions, and equivalent with five tests functions. Again SGO performs superior to GROM in two test functions, worse in three test functions and equivalent with five test functions.

The composite functions are well enough to judge the ability to escape from local minima of a meta-heuristics optimization algorithm. Optimization of composite mathematical functions is a challenging task because only a proper balance between exploration and exploitation allows local optima to be avoided. The results in Table 4 show that none of the algorithms achieve success in finding the global optimal solution. However, LAPO and GROM find a superior solution than then SGO algorithm in solving F24, F25, and F28, whereas SGO finds superior solution then LAPO and GROM in solving F26, F27, and F29 benchmark functions. From Table 4, it is seen that out of six composite test functions according to the WRS test, SGO performs superior to LAPO and GROM in three test functions, worse in three test functions.

In Tables 5 and 6, seven unimodal and six multimodal functions are considered for judging high dimensional parameter optimizations among SGO, LAPO, and GROM algorithm by considering 200 dimensions. Table 5 is for the results of pop_size 200 and 2000 iteration, and Table 6 is for

Table 2 Multimodal benchmark function on 30 repetitions

Functions	LAPO		GROM		SGO
	Results	WRS test	Results	WRS test	Results
F8					
Best	− 1.0613e+4	−	− 9467.296	−	**− 1.0871e+04**
Mean	**− 1.036e+4**		− 9070.8		− 1.0423e+04
Std	1.9994e+3		347.215		949.1964
F9					
Best	**0**	−	**0**	≈	**0**
Mean	1.53344		**0**		**0**
Std	3.70144		**0**		**0**
F10					
Best	9.5009e−9	−	**8.8817e−16**	≈	**8.8818e−16**
Mean	5.8694e−8		**8.8817e−16**		**8.8818e−16**
Std	5.0496e−8		**0**		**0**
F11					
Best	1.7764e−15	−	**0**	≈	**0**
Mean	1.5914e−13		**0**		**0**
Std	2.9758e−13		**0**		**0**
F12					
Best	6.8458e−9	−	3.8742e−9	−	**2.4752e−13**
Mean	0.0104		9.980e−9		**1.8697e−10**
Std	0.0311		5.7890e−9		**6.1247e−10**
F13					
Best	5.5452e−7	−	5.2761e−7	−	**2.7975e−12**
Mean	0.0098		0.003506		**0.00043**
Std	0.0240		0.006430		**0.00067**
Total +		00		00	
Total −		06		03	
Total ≈		00		03	

" −", "+" and "≈" denote that performance of LAPO and GROM is worse, better and similar to SGO, respectively

the results of pop_size 40 and iterations 500. It is clear from both tables that SGO achieves success in finding global optimum on F1, F2, F3, F4, F9, and F11 in both cases. GROM achieves success in finding global optimum on F9 and F11 in both cases. GROM and SGO find equivalent results on F10 for both cases. SGO algorithm finds better results than GROM on F1–F8, F12, and F13 in both cases. In the first case, the LAPO algorithm achieves success on finding global optimum for F9 and F11, it finds equivalent results as of SGO for F10, and worse results on F1–F8, F12 and F13 compared to SGO. However, SGO algorithm finds superior results compared to LAPO for all benchmark functions F1-F13 in the second case. Hence, from Table 5 we summarize that as per the WRS test, SGO performs superior to LAPO and GROM in ten test functions and equivalent on three tests out of thirteen test functions. And from the Table 6, as per the WRS test, SGO performs superior to LAPO in all thirteen test functions and superior to GROM in ten test functions and equivalent with three test functions out thirteen test functions.

Experiment 2

In this experiment, BOA (butterfly optimization algorithm) [29] is compared with the SGO algorithm. For comparison of the performance of algorithms, 30 classical benchmark functions are considered. These benchmark functions are described in appendix B. These functions are chosen from the benchmark set proposed in [127, 128] to determine various features of the algorithm, such as fast convergence, attainment of a large number of local minima points, ability to jump out of local optima and avoid premature convergence. BOA results are taken from paper [29], and for results of the SGO algorithm, the codes are implemented in MATLAB 2016a. Experiments are conducted on an Intel Core i5, 8 GB RAM, and Windows 10 environment.

Table 3 Fixed-dimensional multimodal benchmark function on 30 repetitions

Functions	LAPO		GROM		SGO
	Results	WRS test	Results	WRS test	Results
F14					
Best	**0.9980**	≈	**0.998**	−	**0.9980**
Mean	**0.9980**		1.1964		**0.9980**
Std	5.7495e−8		0.5952		**0**
F15					
Best	**3.0749e−4**	≈	**3.0749e−4**	≈	**3.0749e−04**
Mean	**5.5811e−4**		**5.5811e−4**		**3.0749e−04**
Std	2.2495e−4		2.83762e−19		**2.1112e−15**
F16					
Best	**− 1.0316**	≈	**− 1.0316**	≈	**− 1.0316**
Mean	**− 1.0316**		**− 1.0316**		**− 1.0316**
Std	1.4460e−7		3.315148e−7		**6.7752e−16**
F17					
Best	**0.3979**	−	**0.397887**	≈	**0.3979**
Mean	0.3983		**0.39788**		**0.3979**
Std	4.8405e−4		6.678e−7		**0**
F18					
Best	**3.0000**	≈	2.999	≈	**3.0000**
Mean	**3.0000**		2.999		**3.0000**
Std	**7.5626e−16**		7.02166e−16		**5.2804e−16**
F19					
Best	**− 3.8628**	≈	**− 3.8627**	≈	**− 3.8628**
Mean	**− 3.8628**		**− 3.862**		**− 3.8628**
Std	**8.5422e−16**		**8.88178e−16**		**2.6712e−16**
F20					
Best	− 3.3220	−	− 3.3219	−	**− 3.3220**
Mean	− 3.2729		− 3.29821		**− 3.3220**
Std	0.0571		0.0475558		**1.3897e−15**
F21					
Best	**− 10.1532**	+	**− 10.153**	+	**− 10.1532**
Mean	**− 9.6960**		**− 10.1531**		**−** 5.0552
Std	**0.8042**		**7.944109e−16**		1.0300e−15
F22					
Best	**− 10.4029**	+	**− 10.402**	+	**− 10.4029**
Mean	**− 10.1728**		**− 10.402**		**−** 5.2648
Std	**0.6905**		**1.4862e−15**		0.9704
F23					
Best	**− 10.5364**	+	**− 10.536**	+	**− 10.5364**
Mean	**− 10.2295**		**− 10.53**		**−** 5.3087
Std	**0.6352**		**1.776e−15**		0.9873
Total +		03		03	
Total −		02		02	
Total ≈		05		05	

"−", "+" and "≈" denote that performance of LAPO and GROM is worse, better and similar to SGO respectively

Table 4 Composite benchmark function on 30 repetitions

Functions	LAPO		GROM		SGO
	Results	WRS test	Results	WRS test	Results
F24					
Best	**0**	+	1.3254e−29	+	**0**
Mean	**7.7214e−23**		4.9821e−21		90.0000
Std	**4.2143e−23**		9.8425e−25		73.7865
F25					
Best	2.7340	+	**1.85201**	+	5.8786
Mean	29.87214		**23.75416**		96.5807
Std	32.23484		**19.4826**		91.6926
F26					
Best	116.0883	−	124.5176	−	**6.9055**
Mean	175.86457		165.9840		**109.6482**
Std	90.15781		38.4128		**102.0611**
F27					
Best	265.2541	−	211.48966	−	**100**
Mean	312.145		263.7109		**164.8116**
Std	67.9681		48.9548		**76.3725**
F28					
Best	0	+	**0**	+	18.4288
Mean	45.8928		**34.41452**		122.2605
Std	56.4218		**29.4751**		70.4767
F29					
Best	500.000	−	440.9783	−	**4.4439**
Mean	544.8186		495.41875		**93.0821**
Std	119.6597		98.4126		**113.6061**
Total +		03		03	
Total −		03		03	
Total ≈		00		00	

"−", "+" and "≈" denote that performance of LAPO and GROM is worse, better and similar to SGO respectively

For the BOA algorithm, according to parameter setting in its paper [29], the common control parameter such as pop_size and maximum iterations are 50 and 10,000, respectively. But for the SGO algorithm, we have taken the same pop_size, but maximum iteration is reduced to 500. This is due to our observation of the fast convergence characteristic of SGO. Max_FEs is set to 50,000 ($2 \times 50 \times 500 = 50{,}000$). It is due to two times fitness calculation in one iteration for one particle in population. In this experiment, we have done two tests. In one test, we have set common control parameters the same as above, and in other, we have set pop_size is ten and maximum iteration 500. So Max_FEs is set to 10,000 ($2 \times 10 \times 500 = 10{,}000$). The other specific parameters for each algorithm are given below.

- SGO setting: For SGO, there is only one parameter C called a self-introspection factor. The value of C is empirically set to 0.2.
- BOA settings: Modular modality c is 0.01, and power exponent a is increased from 0.1 to 0.3 throughout iterations, $p = 0.8$. These parameters are set as reported by authors in paper [29].

For each benchmark function, algorithms are run 30 times with different randomly generated populations. Statistical results in terms of mean value, standard-deviation value, the best value, median value, and worse value are reported in tables. Table 9 provides results for test 1 with 50,000 max_FEs, and Table 10 presents results for test 2 with 10,000 max_FEs. For every benchmark function, the best results are boldfaced.

To obtain statistically sound conclusions, Wilcoxon's rank-sum test at a 0.05 significance level is conducted on experimental results of Tables 9, 10 and put in Table 11. The comparison results in terms of best value also are given in

Table 5 Result 200 dimensional with population size 200 and 2000 iterations

Functions	LAPO		GROM		SGO
	Results	WRS test	Results	WRS test	Results
F1					
Best	3.7643e−29	−	4.0197e−283	−	**0**
Mean	8.1296e−29		5.0162e−283		**0**
Std	3.2204e−29		0		**0**
F2					
Best	1.4339e−15	−	1.1623e−156	−	**0**
Mean	1.8098e−15		1.6258e−156		**0**
Std	2.7520e−16		3.2777e−157		**0**
F3					
Best	0.4160	−	1.2908e−207	−	**0**
Mean	2.6872		1.7886e−191		**0**
Std	3.5371		0		**0**
F4					
Best	2.0925e−10	−	6.8726e−122	−	**0**
Mean	7.2703e−10		7.808e−122		**0**
Std	3.3558e−10		7.3555e−123		**0**
F5					
Best	192.6322	−	189.62	−	**183.7878**
Mean	193.0862		190.04		**187.0531**
Std	0.3316		0.30596		**1.3745**
F6					
Best	0.0283	−	0.14347	−	**3.0886e−09**
Mean	0.0412		0.203		**1.4761e−05**
Std	0.0072		0.064899		**3.7908e−05**
F7					
Best	2.8619e−5	−	1.9492e−5	−	**9.1045e−07**
Mean	1.8284e−4		5.4613e−5		**3.8285e−06**
Std	1.3097e−4		1.1139e−6		**1.9758e−06**
F8					
Best	− 5.617e4	−	− 55,179	−	**− 7.1398e+04**
Mean	− 5.5361e4		− 53,752		**− 6.4767e+04**
Std	527.2355		1010.1		**5.9902e+03**
F9					
Best	**0**	≈	**0**	≈	**0**
Mean	**0**		**0**		**0**
Std	**0**		**0**		**0**
F10					
Best	**8.8817e−16**	≈	**8.8817e−16**	≈	**8.8817e−16**
Mean	**8.8817e−16**		**8.8817e−16**		**8.8817e−16**
Std	**0**		**0**		**0**
F11					
Best	**0**	≈	**0**	≈	**0**
Mean	**0**		**0**		**0**
Std	**0**		**0**		**0**
F12					
Best	8.64884e−5	−	1.4254e−5	−	**4.1243e−11**

Table 5 continued

Functions	LAPO		GROM		SGO
	Results	WRS test	Results	WRS test	Results
Mean	1.1944e−4		5.3593e−5		**4.2072e−07**
Std	2.2692e−5		7.2927e−5		**1.3192e−06**
F13					
Best	0.071735	−	0.09547	−	**3.8861e−10**
Mean	0.1748		1.0157		**0.0030**
Std	0.0712		1.4678		**0.0049**
Total +		00		00	
Total −		10		10	
Total ≈		03		03	

"−", "+" and "≈" denote that performance of LAPO and GROM is worse, better and similar to SGO, respectively

Table 11. The last three rows of Table 11 summarize experimental results.

Discussion

In Tables 9 and 10, for the function F22, in the place of best value, worse value, and median value, we have put star '*' because we think that these values are wrongly put in the paper [29]. For the function F28, the minimum function value is given as − 1500 [29]. But the BOA algorithm finds less than that − 1500 and hence a confusion arises on the minimum value. To avoid any conflicts, we have excluded this result and have put '*' in Table 11. From Table 11, we see that except for the F5 function, both Tables 9 and 10 show equivalent results. From Tables 9 and 10, it is clear that SGO algorithm reaches global optimum for 22 benchmark functions such as F1–F4, F6, F8, F11–F13, F15–F24, F26, F27, F29 whereas BOA reaches to global optimum for 18 benchmark functions such as F1–F4, F6, F11–F13, F15–F17, F19, F21–F24, F27, F29. SGO also shows its dominating performance in most functions and satisfactory results in 10 functions such as F7, F8, F9, F10, F14, F18, F20, F25, F26, and F30. For function F26, i.e., shekel 4.5, the BOA algorithm finds superior results than the SGO algorithm. Special attention should be paid to the noisy (quartic) problems as these challenges frequently occur in real-world applications. SGO provided a significant performance boost on this noisy problem and gave an equivalent solution in comparison to BOA, but the best result is better than BOA's best result. It is shown in Table 9. Besides optimization accuracy, convergence speed is quite essential to an optimizer. In this experiment, Table 9 provided the results on 50,000 max_FES with 500 iterations as the termination criterion. Table 10 provided the results on 10,000 max_FES with 500 iterations as the termination, whereas for BOA 10,000 iterations as the termination criterion.

For Test 1, i.e., from Tables 9 and 11, according to the WRS test, it is clear that the SGO algorithm finds better results than the BOA algorithm in nine functions and equivalent results in 19 functions out of 29 benchmark functions. So only in one case, SGO finds worse results than BOA. For test 2, i.e., from Tables 10 and 11, the SGO algorithm performs better than the BOA algorithm in nine functions and equivalent results in 18 functions out of 29 benchmark functions. So only in two cases, the SGO algorithm finds worse results than the BOA algorithm. Similarly, when we compare the results by considering the best results, we find that SGO performs best in ten functions and equally well for 18 functions than BOA in both tests.

From the above experiment and results in discussion, it is found that SGO outperformed than BOA, and the convergence is much fast as it is evident from the maximum numbers of FEs and iterations.

Experiment 3

In this experiment, SSOA (squirrel search optimization algorithm) [30] is compared with the SGO algorithm. For comparison of performances of both the algorithms, 33 benchmark functions are considered. Out of these, 26 functions are classical benchmark functions, and seven are taken from CEC 2014 special session [129]. These benchmark functions are described in Appendix C. These benchmark functions are also described in the paper [30] and taken from [130, 131]. Out of these 26 classical benchmark functions, four are unimodal separable benchmark functions, eight are unimodal non-separable benchmark functions, six are multimodal separable benchmark functions, and eight are multimodal non-separable benchmark functions.

We have directly derived results of SSOA from [30], and for results of the SGO algorithm, the codes are implemented in MATLAB 2016a. Experiments are conducted on Intel Core i5, 8 GB RAM, and Windows 10 environment.

According to parameter settings of SSOA in its respective paper [30], the common control parameter such as pop_size

Table 6 Result 200 dimensional with population size 40 and 500 iterations

Functions	LAPO		GROM		SGO
	Results	WRS test	Results	WRS test	Results
F1					
Best	1.0288e−12	−	3.1809e−65	−	**0**
Mean	1.8090e−9		5.3021e−65		**0**
Std	6.1216e−9		1.6155e−65		**0**
F2					
Best	2.0087e−7	−	3.7697e−36	−	**0**
Mean	3.5023e−6		5.2158e−36		**0**
Std	4.3034e−6		1.0254e−36		**0**
F3					
Best	0.9660	−	1.1586e−62	−	**0**
Mean	6.7051		2.8633e−57		**0**
Std	5.7391		2.553e−57		**0**
F4					
Best	9.8895e−6	−	3.8682e−30	−	**0**
Mean	2.4845e−5		4.0382e−30		**0**
Std	1.4955e−5		1.4237e−31		**0**
F5					
Best	196.3227	−	195.44	−	**194.5407**
Mean	197.3273		197.6		**195.2793**
Std	0.5454		0.55796		**0.4570**
F6					
Best	13.7919	−	16.756	−	**0.0345**
Mean	16.5038		17.942		**0.2781**
Std	1.3669		1.1032		**0.1591**
F7					
Best	3.5309e−4	−	1.7976e−4	−	**5.1821e−07**
Mean	0.0010		2.8908e−4		**2.6497e−06**
Std	4.5410e−4		1.1115e−4		**1.5043e−05**
F8					
Best	− 1.5279e4	−	− 15,765	−	**− 6.5602e+04**
Mean	**−** 1.324e4		− 15,167		**− 5.4946e+04**
Std	815.8747		430.67		**5.4972e+03**
F9					
Best	4.5475e−13	−	**0**	≈	**0**
Mean	9.2776e−10		**0**		**0**
Std	3.6949e−10		**0**		**0**
F10					
Best	7.6647e−8	−	**8.8818e−16**	≈	**8.8818e−16**
Mean	4.2081e−6		**8.8818e−16**		**8.8818e−16**
Std	2.8691e−6		**0**		**0**
F11					
Best	2.8019e−12	−	**0**	≈	**0**
Mean	5.6723e−10		**0**		**0**
Std	2.8363e−10		**0**		**0**
F12					
Best	0.0995	−	0.15496	−	**4.0511e−04**

Table 6 continued

Functions	LAPO		GROM		SGO
	Results	WRS test	Results	WRS test	Results
Mean	0.0149		0.1663		**0.0025**
Std	0.1178		0.00805		**0.0020**
F13					
Best	12.7156	−	12.803	−	**0.0177**
Mean	1.9412		12.822		**0.3140**
Std	15.7881		0.024253		**0.3987**
Total +		00		00	
Total −		13		10	
Total ≈		00		03	

" −", "+" and "≈" denote that performance of LAPO and GROM is worse, better and similar to SGO, respectively

Table 7 Overall algorithms comparison on Wilcoxon's rank-sum test using Tables 1, 2, 3 and 4

Functions	Unimodal		Multimodal		Fixed dimensional		Composite		Overall in LAPO	Overall in GROM
	LAPO	GROM	LAPO	GROM	LAPO	GROM	LAPO	GROM		
Total +	01	01	00	00	03	03	03	03	7	07
Total −	06	06	06	03	02	02	03	03	17	14
Total ≈	00	00	00	03	05	05	00	00	05	08
Total	07	07	06	06	10	10	06	06	29	29
	07		06		10		06		29	

Table 8 Overall algorithms comparison of Wilcoxon's rank-sum test using Tables 5 and 6

Functions	Test of 200 dimensional with pop_size 200 and maximum iteration 2000		Test of 200 dimensional with pop_size 40 and maximum iteration 500		Overall in LAPO	Overall in GROM
	LAPO	GROM	LAPO	GROM		
Total +	00	00	00	00	00	00
Total −	10	10	13	10	23	20
Total ≈	03	03	00	03	03	06
Total	13	13	13	13	26	26
	13		13		26	

is 50, and the maximum iteration is 500. For seven CEC2014 benchmark functions, pop_size is 50, and the maximum iteration is 6000. So max_FEs is 300,000. So for the SGO algorithm, we have taken pop_size is 25, and the maximum iteration is 500. Hence max_FEs are 25,000($2 \times 25 \times 500 = 25{,}000$, two is due to two-time fitness calculation in one iteration for one particle in population) for classical benchmark functions, and pop_size is 50, and maximum iteration is 3000. So max_FEs is 300,000 ($2 \times 50 \times 3000 = 300{,}000$) for CEC2014 benchmark functions.

The other specific parameters for each algorithm are given below.

- SGO setting: For SGO, there is only one parameter C called a self-introspection factor. The value of C is empirically set to 0.2.

SOA settings: nutritious food resources $N_{fs} = 4$, gliding constant $G_c = 1.9$ and predator presence probability $P_{dp} = 0.1$. Parameters are set as reported by authors in paper [30].

For each benchmark function, algorithms are run 30 times with different randomly generated populations. Statistical results in terms of mean value, corresponding standard-deviation, the best value, and worse value are reported in tables. Table 12 indicates the test results for unimodal separable benchmark functions. Table 13 reports the test results for unimodal non-separable benchmark functions. Table 14

Table 9 Comparison on BOA and SGO on 30 independent runs with 50,000 fitness function evaluations

Fun no.	Function Name	Algorithms	Mean	Std	Best	Median	Worse
F1	Sphere	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F2	Beale	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F3	Cigar	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F4	Step	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F5	Quartic function with noise	BOA	3.8917e−05	2.9003e−05	5.8800e−05	3.5850e−05	1.2000e−05
		SGO	**2.4890e−05**	1.0695e−05	**4.0050e−06**	4.6955e−05	4.2739e−05
F6	Bohachevsky	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F7	Ackley	BOA	1.7183e+00	0	1.7183e+00	1.7183e+00	1.7183e+00
		SGO	**8.8818e−16**	**0**	**8.8818e−16**	**8.8818e−16**	**8.8818e−16**
F8	Griewank	BOA	1.8472e−19	2.6886e−20	1.6300e−19	1.6300e−19	2.1700e−19
		SGO	**0**	**0**	**0**	**0**	**0**
F9	Levy	BOA	4.4108e−01	5.7467e−02	2.8900e−01	3.7800e−01	5.7400e−01
		SGO	**4.5472e−04**	**2.3885e−04**	**7.2873e−05**	**4.7006e−04**	**8.7819e−04**
F10	Michalewiz	BOA	− 5.3382	− 5.6092	− 5.7700	− 3.6200	− 2.2500
		SGO	**− 9.1418**	**0.2639**	**− 9.6141**	**− 9.6141**	**− 8.6585**
F11	Rastrigin	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F12	Alpine	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F13	Schaffer	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F14	Rosenbrock	BOA	2.8837e+01	3.1281e−02	2.8754e+01	2.8754e+01	2.8927e+01
		SGO	**23.2241**	**0.7375**	**21.7366**	**23.2401**	**24.5057**
F15	Easom	BOA	**− 1.0000**	**0**	**− 1.0000**	**− 1.0000**	**− 1.0000**
		SGO	**− 1**	**0**	**− 1**	**− 1**	**− 1**
F16	Shubert	BOA	**− 1.8673e+02**	**2.06493e−11**	**− 1.8673e+02**	**− 1.8673e+02**	**− 1.8673e+02**
		SGO	**− 186.7309**	**4.7915e−14**	**− 186.7309**	**− 186.7309**	**− 186.7309**
F17	Schwefel 1.2	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F18	Schwefel 2.21	BOA	6.9906e−153	1.4788e−152	3.7983e−155	2.3793e−153	9.7687e−152
		SGO	**0**	**0**	**0**	**0**	**0**
F19	Schwefel 2.22	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F20	Schwefel 2.26	BOA	− 2.2662e+03	4.5626e+02	− 2.8790e+03	− 2.3458e+03	− 1.8373e+03
		SGO	**− 9.3595e+03**	**1.0556e+03**	**− 1.1721e+04**	**− 9.5886e+03**	**− 7.9691e+03**
F21	Booth	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F22	Goldstein price	BOA	3.0000e+00	0.0000e+00	0.0000e+00*	0.0000e+00*	0.0000e+00*
		SGO	**3.0000**	**0**	**3.0000**	**3.0000**	**3.0000**
F23	Matyas	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**

Table 9 continued

Fun no.	Function Name	Algorithms	Mean	Std	Best	Median	Worse
F24	Powell	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F25	Power sum	BOA	2.8400e−02	1.3179e−02	8.9600e−03	2.6800e−02	5.4300e−02
		SGO	**7.2637e−04**	**5.4052e−04**	**5.6591e−05**	**6.4722e−04**	**0.0020**
F26	Shekel 4.5	BOA	− **10.200**	0	**− 10.200**	− 10.200	− 10.200
		SGO	− 5.3101	1.1399	**− 10.1532**	− 5.0552	− 5.0552
F27	Sum squares	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F28	Trid	BOA	− 2.7500e+07*	5.6500e+07*	− 9.6100e+07*	− 3.1800e+07*	7.3000e+07*
		SGO	**− 3.6951e+03**	340.1588	− 4.1038e+03	− 3.8835e+03	− 3.0904e+03
F29	Zettl	BOA	− **3.7900e−03**	**0**	**− 3.7900e−03**	**− 3.7900e−03**	− **3.7900e−03**
		SGO	− **3.7900e−03**	**0**	**− 3.7900e−03**	**− 3.7900e−03**	− **3.7900e−03**
F30	Leon	BOA	1.1527e−06	9.4711e−07	1.0700e−08	6.9200e−07	2.6800e−06
		SGO	**2.4663e−12**	**3.3155e−12**	**1.3378e−13**	**5.9250e−13**	**9.5362e−12**

reports the test results for multimodal separable benchmark functions. Table 15 reports the test results for multimodal non-separable benchmark functions, and Table 16 reports the test results for CEC2014 functions. For every benchmark function, the best results are boldfaced.

To obtain statistically sound conclusions, the WRS test at a 0.05 significance level is conducted on the experimental results of Tables 12, 13, 14, 15, 16 and reported in Table 17. A comparison of the best results obtained in Tables 12, 13, 14, 15, 16, also published in Table 17. The last three rows of Table 17 summarize experimental results.

Discussion

It is clear from Table 12 that the SGO achieves success in finding global optimum on unimodal separable functions F1, F2, and F3. For F1, the performance of the SSOA is found identical to the SGO. Only for F4, the SGO could not reach the global optimum region but find better results than the SSOA.

Table 13 provides results on unimodal non-separable functions. It is clear from the table that the SGO achieves success in finding global optimum on functions F5, F6, F7, F9, F10, F11, but the SSOA achieves success in finding global optimum only on F6. For F6 only, the performance of SSOA is found identical to the SGO. Only for F8 and F12, SGO could not reach the global optimum region. For the F8 function, SSOA finds a better result than SGO, and for F12 SGO, find a better result than SSOA.

It is clear from Table 14 that the SGO achieves success in finding global optimum on multimodal separable functions F13, F14, F15, F18, but SSOA achieves success in finding only on F13 and F15. For F13 and F15, the performance of SSOA is found identical to SGO. For F16 and F17, SGO could not reach the global optimum region but find better results than the SSOA.

It is clear from Table 15 results that the SGO achieves success in finding global optimum on multimodal non-separable functions F19, F20, F21, F22, F23, F25, but the SSOA achieves success in finding only on F20, F21, F22, and F23. Only for F24 and F26, SGO could not reach the global optimum region. For F24, SSOA finds better results, and for F26, SGO finds a better result.

It is clear from Table 16 that SGO finds a better solution in F27, F28, F29, then SSOA, whereas SSOA finds better in F30 and F31 then SGO, and in F32 and F33, both SGO and SSOA finds equivalent results.

From Table 17, according to the WRS test, we find that the SGO algorithm performs superior solutions in three cases, equivalent solution in one case out of four unimodal separable benchmark functions than the SSOA algorithm. Out of eight unimodal non-separable benchmark functions, the SGO algorithm performs superior solutions in six cases, equivalent solution in one case and worse solution in one case than the SSOA algorithm. Out of six multimodal separable benchmark functions, the SGO algorithm performs superior solutions in four cases, an equivalent solution in two cases than the SSOA algorithm. Out of eight multimodal non-separable benchmark functions, the SGO algorithm performs superior solutions in three cases, equivalent solution in four cases, worse solution in one case than the SSOA algorithm. Out of seven CEC2014 benchmark functions, the SGO algorithm performs superior solutions in three cases, equivalent solutions in two cases, and worse solutions in two cases than the SSOA algorithm.

Table 10 Comparison of BOA and SGO on 30 independent runs with 10,000 fitness function evaluations

Fun no.	Function name	Algorithms	Mean	Std	Best	Median	Worse
F1	Sphere	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F2	Beale	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F3	Cigar	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F4	Step	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F5	Quartic function with noise	BOA	**3.8917e−05**	2.9003e−05	5.8800e−05	3.5850e−05	1.2000e−05
		SGO	1.0432e−04	7.9660e−05	**8.1471e−06**	8.8846e−05	3.1340e−04
F6	Bohachevsky	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F7	Ackley	BOA	1.7183e+00	0	1.7183e+00	1.7183e+00	1.7183e+00
		SGO	**8.8818e−16**	**0**	**8.8818e−16**	**8.8818e−16**	**8.8818e−16**
F8	Griewank	BOA	1.8472e−19	2.6886e−20	1.6300e−19	1.6300e−19	2.1700e−19
		SGO	**0**	**0**	**0**	**0**	**0**
F9	Levy	BOA	4.4108e−01	5.7467e−02	2.8900e−01	3.7800e−01	5.7400e−01
		SGO	**0.0016**	**7.9751e−04**	**4.5448e−04**	**0.0013**	**0.0032**
F10	Michalewiz	BOA	− 5.3382	− 5.6092	− 5.7700	− 3.6200	− 2.2500
		SGO	**− 8.1452**	**0.6628**	**− 9.3669**	**− 8.1402**	**− 6.8557**
F11	Rastrigin	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F12	Alpine	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F13	Schaffer	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F14	Rosenbrock	BOA	2.8837e+01	3.1281e−02	2.8754e+01	2.8754e+01	2.8927e+01
		SGO	**28.3809**	0.3189	**27.6996**	28.4405	28.7686
F15	Easom	BOA	**− 1.0000**	**0**	**− 1.0000**	**− 1.0000**	**− 1.0000**
		SGO	**− 1**	**0**	**− 1**	**− 1**	**− 1**
F16	Shubert	BOA	**− 1.8673e+02**	**2.06493e−11**	**− 1.8673e+02**	**− 1.8673e+02**	**− 1.8673e+02**
		SGO	**− 186.7309**	**4.7915e−11**	**− 186.7309**	**− 186.7309**	**− 186.7309**
F17	Schwefel 1.2	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F18	Schwefel 2.21	BOA	6.9906e−153	1.4788e−152	3.7983e−155	2.3793e−153	9.7687e−152
		SGO	**0**	**0**	**0**	**0**	**0**
F19	Schwefel 2.22	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F20	Schwefel 2.26	BOA	**−** 2.2662e+03	4.5626e+02	**−** 2.8790e+03	**−** 2.3458e+03	**−** 1.8373e+03
		SGO	**− 6.9949e+03**	**1.0556e+03**	**− 9.3437e+03**	− **0.6536e+03**	**− 5.6009e+03**
F21	Booth	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F22	Goldstein price	BOA	**3.0000**	**0**	**3.0000**	**3.0000**	**3.0000**
		SGO	**3.0000**	**0**	**3.0000**	**3.0000**	**3.0000**
F23	Matyas	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**

Table 10 continued

Fun no.	Function name	Algorithms	Mean	Std	Best	Median	Worse
F24	Powell	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F25	Power sum	BOA	2.8400e−02	1.3179e−02	8.9600e−03	2.6800e−02	5.4300e−02
		SGO	**0.0012**	**0.0014**	**3.5053e−05**	**3.0905e−04**	**0.0045**
F26	Shekel 4.5	BOA	− **10.2000**	0	**− 10.2000**	− 10.2000	− 10.2000
		SGO	− 5.3101	1.1399	**− 10.1532**	− 5.0552	-5.0552
F27	Sum squares	BOA	**0**	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**	**0**
F28	Trid	BOA	− 2.7500e+07*	5.6500e+07*	− 9.6100e+07*	− 3.1800e+07*	7.3000e+07*
		SGO	− 286.1084	448.9151	− 1.4262e+03	− 99.4426	–24.0904
F29	Zettl	BOA	**− 3.7900e−03**	**0**	**− 3.7900e−03**	**− 3.7900e−03**	**− 3.7900e−03**
		SGO	**− 3.7912e−03**	**1.3348e−18**	**− 3.7912e−03**	**− 3.7912e−03**	**− 3.7912e−03**
F30	Leon	BOA	1.1527e−06	9.4711e−07	1.0700e−08	6.9200e−07	2.6800e−06
		SGO	**2.4663e−09**	**3.3155e−07**	**1.3378e−12**	**5.9250e−09**	**9.5362e−7**

While comparing in terms of best solution value between SGO and SSOA algorithm, then the SGO algorithm performs better in three cases and similar to SSOA algorithm in one case, out of four unimodal separable benchmark functions. The SGO algorithm is better performing in six cases, one case being similar, and one case giving worse solution than SSOA algorithm out of eight unimodal non-separable benchmark functions. The SGO algorithm is better in two cases and similar to the SSOA algorithm in four cases, out of six multimodal separable benchmark functions. The SGO algorithm is better in one case, similar in six cases and gives worse solution than SSOA algorithm in one case, out of eight multimodal non-separable benchmark function. The SGO algorithm is better in three cases and gives similar solution with SSOA algorithm in four cases, out of seven CEC2014 benchmark functions.

Experiment 4

In this experiment, the VPL (volleyball premier league) algorithm [60] is compared with the SGO algorithm. For performance comparison of algorithms, 23 classical benchmark-functions are considered. Out of which seven are unimodal benchmark-functions, six are multimodal benchmark-functions, ten are fixed-dimension multimodal benchmark-functions. These benchmark-functions are described in Appendix A.

We have directly derived results of the VPL algorithm from [60], and for results of the SGO algorithm, the codes are implemented in MATLAB 2016a. Experiments are conducted on an Intel Core i5, 8 GB memory laptop in Windows 10 environment.

According to parameter settings of the VPL algorithm in its respective paper [60], the common control parameter, such as max_FEs, is 100,000. So, for the SGO algorithm, common control parameters such as pop_size are set to 50, and maximum iteration is set to 1000. So max_FEs is $2 \times 50 \times 1000 = 100{,}000$. The other specific parameters for each algorithm are given below.

- SGO setting: For SGO, there is only one parameter C called a self-introspection factor. The value of C is empirically set to 0.2.
- VPL setting: For VPL $L = 60$, PC $= 0.5$, rate of promoted $\delta_{Pr} = 0.05$, $\delta_{st} = 0.54$, $N = 10$, and $\beta = 7$.

Parameters are set as reported by authors in paper [60].

For each benchmark-function, algorithms are run 30 times with different randomly generated populations. Statistical results in terms of mean value, corresponding standard-deviation, the best value, and worse value are reported in tables. Table 18 reports the test results for unimodal benchmark functions. The test results for multimodal benchmark functions are reported in Table 19. Table 20 reports the test result for fixed dimensional multimodal benchmark functions. For every benchmark function, the best results are boldfaced.

To obtain statistically sound conclusions, the WRS test at a 0.05 significance level is conducted on the experimental results of Tables 18, 19, 20 and reported in their respective tables. Also, comparisons on the best results are obtained and are given in the respective table also. The last three rows of tables summarize experimental results.

According to Table 18, SGO has gained the best performance and consistently performed better than VPL algo-

Table 11 Wilcoxon rank test result and the comparison result on the best value between BOA and SGO

function	Function Name	Wilcoxon rank test result using Table 9	Comparison of best result of Table 9	Wilcoxon rank test result using Table 10	Comparison of best result of Table 10
F1	Sphere	≈	Same	≈	Same
F2	Beale	≈	Same	≈	Same
F3	Cigar	≈	Same	≈	Same
F4	Step	≈	Same	≈	Same
F5	Quartic function with noise	≈	Best	+	Best
F6	Bohachevsky	≈	Same	≈	Same
F7	Ackley	−	Best	−	Best
F8	Griewank	−	Best	−	Best
F9	Levy	−	Best	−	Best
F10	Michalewiz	−	Best	−	Best
F11	Rastrigin	≈	Same	≈	Same
F12	Alpine	≈	Same	≈	Same
F13	Schaffer	≈	Same	≈	Same
F14	Rosenbrock	−	Best	−	Best
F15	Easom	≈	Same	≈	Same
F16	Shubert	≈	Same	≈	Same
F17	Schwefel 1.2	≈	Same	≈	Same
F18	Schwefel 2.21	−	Best	−	Best
F19	Schwefel 2.22	≈	Same	≈	Same
F20	Schwefel 2.26	−	Best	−	Best
F21	Booth	≈	Same	≈	Same
F22	Goldstein price	≈	Same	≈	Same
F23	Matyas	≈	Same	≈	Same
F24	Powell	≈	Same	≈	Same
F25	Power sum	−	Best	−	Best
F26	Shekel 4.5	+	Worse	+	Worse
F27	Sum squares	≈	Same	≈	Same
F28	Trid	*	*	*	*
F29	Zettl	≈	Same	≈	Same
F30	Leon	−	Best	−	Best
Total +		01	1, worse	02	1, worse
Total −		09	10, best	09	10, best
Total ≈		19	18, same	18	18, same

"−", "+" and "≈" denote that performance of BOA is worse, better and similar to SGO respectively

Table 12 Comparison on SSOA and SGO on 30 independent runs on unimodal separable benchmark functions

Function no.	Function name	Algorithms	Best	Worse	Mean	Std
F1	Step	SSOA	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**
F2	Sphere	SSOA	7.9225e−20	5.7411e−07	4.1689e−08	1.4356e−07
		SGO	**0**	**0**	**0**	**0**
F3	Sumsquares	SSOA	2.0052e−28	2.3194e−06	1.5201e−07	4.6741e−07
		SGO	**0**	**0**	**0**	**0**
F4	Quartic	SSOA	3.0998e−02	9.9258e−01	5.0192e−01	2.9565e−01
		SGO	**2.5878e−06**	**8.2220e−05**	**3.6441e−05**	**2.5849e−05**

Table 13 Comparison on SSOA and SGO on 30 independent runs on unimodal nonseparable benchmark functions

Function no.	Function name	Algorithms	Best	Worse	Mean	Std
F5	Beale	SSOA	2.1832e−29	2.7633e−20	9.5584e−22	5.0400e−21
		SGO	**0**	**0**	**0**	**0**
F6	Easom	SSOA	**− 1**	**− 1**	**− 1**	**0**
		SGO	**− 1**	**− 1**	**− 1**	**0**
F7	Matyas	SSOA	1.5111e−29	2.0707e−24	1.542e−25	4.7571e−25
		SGO	**0**	**0**	**0**	**0**
F8	Colville	SSOA	**8.5561e−21**	**2.4871e−08**	**1.4309e−09**	**4.6907e−09**
		SGO	4.2541e−11	6.9093e−05	8.4402e−06	1.8988e−05
F9	Zakharov	SSOA	1.9954e−23	1.5225e−07	5.2215e−09	2.7772e−08
		SGO	**0**	**0**	**0**	**0**
F10	Schwefel 2.22	SSOA	2.2266e−08	7.1423e−03	5.1849e−04	1.4144e−03
		SGO	**0**	**0**	**0**	**0**
F11	Schwefel 1.2	SSOA	6.6803e−18	3.4907e−04	1.6925e−05	6.6811e−05
		SGO	**0**	**0**	**0**	**0**
F12	Dixon-Price	SSOA	1.8308e−01	6.6951e−01	2.2412e−01	1.2107e−01
		SGO	**3.3801e−04**	**0.1258**	**0.0946**	**0.0731**

Table 14 Comparison on SSOA and SGO on 30 independent runs on multimodal separable benchmark-functions

Function no.	Function name	Algorithms	Best	Worse	Mean	Std
F13	Bohachevsky1	SSOA	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**
F14	Booth	SSOA	1.2622e−29	8.1255e−24	9.5859e−25	1.7996e−24
		SGO	**0**	**0**	**0**	**0**
F15	Michalewicz2	SSOA	**− 1.8013**	**− 1.8013**	**− 1.8013**	**1.0275e−15**
		SGO	**− 1.8013**	**− 1.8013**	**− 1.8013**	**6.8344e−16**
F16	Michalewicz5	SSOA	**− 4.6877**	− 3.5563	− 4.3479	3.2785e−01
		SGO	**− 4.6877**	**−** 4.4948	**− 4.5862**	0.0848
F17	Michalewicz10	SSOA	− 9.4806	− 5.9146	− 7.5900	9.9570e−01
		SGO	**− 9.5515**	**− 8.4770**	**− 9.0513**	**0.3087**
F18	Rastrigin	SSOA	0	7.6657e−06	4.9059e−07	1.5057e−06
		SGO	**0**	**0**	**0**	**0**

rithms. SGO has an excellent performance in exploitation and convergence then VPL, and it successfully overcomes to solve all the problems within this category. It is clear from the results that SGO achieves success in finding global optimum on F1–F4. For F5, F6, and F7, the performance of SGO is better than VPL.

From Table 18, we find that in all the cases of unimodal benchmark functions according to the WRS test, the SGO algorithm shows better performance than the VPL algorithm. Similarly, comparison on best results obtained by algorithms, we find that the SGO algorithm gets either the best result than the VPL algorithm or similar result with the VLP algorithm. From the table, it is evident that the SGO algorithm outperforms in solving unimodal benchmark functions in comparison to VPL algorithms.

In Table 19, there is '*' mark in the first row with the result of the VPL algorithm to say that the result may be put wrongly as the minimum value of the F8 function is − 12,569.487. But we get less than that in the paper and hence a confusion arising on the minimum value. To avoid any conflicts, we have excluded this result and have put '*' in Table 19. So, for comparison, we have considered only five multimodal functions. The multimodal functions are beneficial, while the exploration capability of the optimization algorithm is considered. Form Table 19, results show that the SGO algorithm is eligible for solving problems with challenging search space. The table shows that SGO has consistently performed better than VPL algorithms. SGO has an excellent performance in exploration and convergence, and it successfully overcomes to solve all the problems within this category. It is clear from table results that both SGO and VPL achieve

Table 15 Comparison on SSOA and SGO on 30 independent runs on multimodal nonseparable functions

Function no.	Function name	Algorithms	Best	Worse	Mean	Std
F19	Schaffer	SSOA	0	9.7159e−03	9.7159e−04	2.9646e−03
		SGO	**0**	**0**	**0**	**0**
F20	Six Hump Camel Back	SSOA	**− 1.03163**	**− 1.03163**	**− 1.03163**	**4.5168e−16**
		SGO	**− 1.03163**	**− 1.03163**	**− 1.03163**	**0**
F21	Boachevsky2	SSOA	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**
F22	Boachevsky3	SSOA	**0**	**0**	**0**	**0**
		SGO	**0**	**0**	**0**	**0**
F23	Shubert	SSOA	**− 186.73**	**− 186.73**	**− 186.73**	**2.6389e−14**
		SGO	**− 186.73**	**− 186.73**	**− 186.73**	**1.6722e−15**
F24	Rosenbrock	SSOA	**3.9637e−19**	**2.8475e+01**	**9.4919e−01**	**5.1988e+00**
		SGO	25.4955	26.6390	26.1690	0.3605
F25	Griewank	SSOA	**0**	4.1375e−05	3.435e−06	9.6702e−06
		SGO	**0**	**0**	**0**	**0**
F26	Ackley	SSOA	2.2418e−10	2.5867e−03	1.3915e−04	4.8513e−04
		SGO	**− 8.8818e−16**	**− 8.8818e−16**	**− 8.8818e−16**	**0**

Table 16 Comparison on SSOA and SGO on 30 independent runs on CEC 2014 benchmark-functions with 30 dimension

Function no.	Function name	Algorithms	Best	Worse	Mean	Std
F27	CEC1	SSOA	9.1044e+04	1.7503e+06	8.1899e+05	4.0164e+05
		SGO	**3.7329e+04**	3.6134e+05	**1.5721e+05**	9.9594e+04
F28	CEC2	SSOA	2.1599e+02	2.8185e+04	1.0049e+04	9.8268e+03
		SGO	**200.0005**	635.8838	**331.9643**	119.1971
F29	CEC4	SSOA	**4.0000e+02**	5.4414e+02	4.5717e+02	3.9877e+01
		SGO	**400.0035**	463.4008	**402.8456**	11.5184
F30	CEC17	SSOA	7.0506e+03	6.0131e+04	**2.6151e+04**	1.5238e+04
		SGO	**5.9331e+03**	8.3743e+04	3.2450e+04	1.9765e+04
F31	CEC23	SSOA	**2.500e+03**	2.500e+03	**2.500e+03**	8.4083e−10
		SGO	**2500**	2.6152e+03	2.5576e+03	58.6071
F32	CEC24	SSOA	**2.600e+03**	**2.600e+03**	**2.600e+03**	8.3747e−02
		SGO	**2600**	**2600**	**2600**	0
F33	CEC125	SSOA	**2.700e+03**	**2.700e+03**	**2.700e+03**	4.8285e−11
		SGO	**2700**	**2700**	**2700**	0

success in finding global optimum on F9 and F1. For F10, both SGO and VPL get equivalent results. For F12 and F13, the SGO algorithm finds better results than the VPL algorithm. According to the WRS test from Table 19, we find that in two out of five cases, SGO algorithm shows better performance than the VPL algorithm and similar in three out of five cases with the VPL algorithm. Similarly, comparison on best results obtained by algorithms, we get that the SGO algorithm either gets the best result than the VPL algorithm or similar result to the VPL algorithm. Hence, we find that the SGO algorithm shows best performance in solving multimodal benchmark functions in comparison to VPL algorithms.

In Table 20, there is '*' mark with the result of the VPL algorithm to say that there, the result might be put wrongly as the minimum value of the F15 function is 3.0749e−04. But we get different values in the paper and hence a confusion arising on minimum value. To avoid any conflicts, we have excluded this result and have put '*' in Table 20. The fixed-dimensional multimodal functions are designed to have many local optimal where computation complexity increases drastically with the problem size. The results reported from

Table 17 Wilcoxon rank test result and comparison result on best value between SSOA and SGO

Function no.	Wilcoxon test	Comparison on best	Function no.	Wilcoxon test	Comparison on best	Function no.	Wilcoxon test	Comparison on best	Function no.	Wilcoxon test	Comparison on best	Function no.	Wilcoxon test	Comparison on best
F1	≈	Same	F5	−	Best	F13	≈	Same	F19	−	Same	F27	−	Best
F2	−	Best	F6	≈	Same	F14	−	Best	F20	≈	Same	F28	−	Best
F3	−	Best	F7	−	Best	F15	≈	Same	F21	≈	Same	F29	−	Same
F4	−	Best	F8	+	Worse	F16	−	Same	F22	≈	Same	F30	+	Best
			F9	−	Best	F17	−	Best	F23	≈	Same	F31	+	Same
			F10	−	Best	F18	−	Same	F24	+	Worse	F32	≈	Same
			F11	−	Best				F25	−	Same	F33	≈	Same
			F12	−	Best				F26	−	Best			
T −	03	S 1		06	S 1		04	S 4		03	S 6		03	S 4
T +	00	B 3		01	B 6		00	B 2		01	B 1		02	B 3
T ≈	01	W 0		01	W 1		02	W 0		04	W 1		02	W 0

" −", "+" "≈" denote that performance of SSOA is worse, better and similar to SGO respectively and same, best, and worse are similar, better and worse solutions of SGO then SSOA algorithm in term of the best result

Table 18 Result of unimodal benchmark functions

Function no.	Algorithms	Best	Comparison on best result	Worse	Mean	Std	WRS test
F1	VPL	**0.00e+00**	Same	2.34e−130	7.81e−132	4.20e−131	−
	SGO	**0.00e+00**		0.00e+00	**0.00e+00**	0.00e+00	
F2	VPL	1.12e−102	Best	2.85e−89	1.13e−90	5.13e−90	−
	SGO	**0.00e+00**		0.00e+00	**0.00e+00**	0.00e+00	
F3	VPL	1.93e−33	Best	1.53e−02	8.16e−04	2.85e−03	−
	SGO	**0.00e+00**		0.00e+00	**0.00e+00**	0.00e+00	
F4	VPL	**0.00e+00**	Same	1.63e−28	1.54e−29	3.96e−29	−
	SGO	**0.00e+00**		0.00e+00	**0.00e+00**	0.00e+00	
F5	VPL	2.58e+01	Best	2.67e+01	2.62e+01	2.76e−01	−
	SGO	**17.8419**		21.6899	**20.4626**	1.0124	
F6	VPL	1.82e−05	Best	2.34e−03	4.09e−04	5.33e−04	−
	SGO	**1.0720e−26**		2.9247e−18	**1.3337e−19**	5.4652e−19	
F7	VPL	4.67e−05	Best	4.81e−03	1.93e−03	1.36e−03	−
	SGO	**1.6238e−07**		2.3695e−05	**9.9687e−06**	7.6441e−06	
Total no of the best		05			Total +	00	
Total no. of worse		00			Total −	00	
Total no. of the same		02			Total ≈	07	

" −", "+","≈" denote performance of VPL is worse, better and similar to SGO respectively and same, best, and worse are the similar, better, and worse solutions of SGO then SSA algorithm in terms of the best result

Table 20 shows that the SGO algorithm achieves success in finding global optimum on F14, F15, F16, F17, F18, and F19.

In contrast, VPL reaches optimal solution only for F14, F16, F17, and F18. For shekel family, i.e., for F21, F22 and F23 VPL finds better result than SGO algorithm. From Table 20, according to the WRS, we find that out of ten cases, SGO algorithm shows better performance in three cases and similar performance in four cases than the VPL algorithm In contrast, the VPL algorithm shows its better performance in three cases out of ten than the SGO algorithm. Similarly, comparison on best results obtained by algorithms, we find that the SGO algorithm either gets the best result than the VPL algorithm or similar result with the VLP algorithm except for two cases. Hence, we can see that the VPL algorithm shows best performance in solving fixed dimensional multimodal benchmark functions in comparison to SGO algorithms.

Experiment 5

In this experiment, SELO (socio evolution and learning optimization) [61] algorithm is compared with the SGO algorithm. For comparison of the performance of algorithms, a set of 50 benchmark functions are considered. These set of test functions include problems of varying complexity levels,

Table 19 Result of multimodal benchmark functions

Function no.	Algorithms	Best	Comparison on best result	Worse	Mean	Std	WRS test
F8	VPL	− 1.19e+112*	–	− 7.38e+90*	− 4.68e+110*	2.15e+111*	–
	SGO	− 1.1385e+04		− 7.7527e+03	**− 9.0646e+03**	1.0705e+03	
F9	VPL	**0.00e+00**	Same	0.00e+00	**0.00e+00**	0.00e+00	≈
	SGO	**0.00e+00**		0.00e+00	**0.00e+00**	0.00e+00	
F10	VPL	**8.88e−16**	Same	8.88e−16	**8.88e−16**	9.86e−32	≈
	SGO	**8.8818e−16**		8.8818e−16	**8.8818e−16**	0.00e+00	
F11	VPL	**0.00e+00**	Same	0.00e+00	**0.00e+00**	0.00e+00	≈
	SGO	**0.00e+00**		0.00e+00	**0.00e+00**	0.00e+00	
F12	VPL	1.11e−06	Best	6.56e−05	2.58e−05	1.74e−05	−
	SGO	**3.6451e−26**		5.6154e−23	**7.3910e−24**	1.3826e−23	
F13	VPL	2.63e−05	Best	2.31e−03	4.18e−04	4.84e−04	−
	SGO	**2.5068e−25**		0.0110	**6.8145e−12**	2.2671e−08	
Total no of the best			02			Total +	01
Total no. of worse			01			Total −	02
Total no. of the same			03			Total ≈	03

" −", "+", "≈" denote performance of VPL is worse, better, and similar to SGO respectively and same, best and worse are the similar, better and worse solution of SGO then VPL algorithm in term of the best result

Table 20 Results on fixed-dimensional benchmark functions

Function no.	Algorithms	Best	Comparison on best result	Worse	Mean	Std	WRS test
F14	VPL	**9.98e−01**	Same	9.98e−01	**9.98e−01**	2.32e−13	≈
	SGO	**0.9980**		0.9980	**0.9980**	5,3418e−15	
F15	VPL	2.45e−05*	Best	1.82e−03	1.25e−03	3.08e−04	−
	SGO	**3.0749e−04**		3.0749e−04	**3.0749e−04**	1.2533e−19	
F16	VPL	**− 1.03e+00**	Same	− 1.03e+00	**− 1.03e+00**	2.56e−06	≈
	SGO	**− 1.0316**		− 1.0316	**− 1.0316**	6.7752e−16	
F17	VPL	**3.98e−01**	Same	3.98e−01	**3.98e−01**	2.69e−06	≈
	SGO	**0.3979**		0.3979	**0.3979**	4.1563e−15	
F18	VPL	**3.00e+00**	Same	3.00e+00	**3.00e+00**	7.58e−05	≈
	SGO	**3.0000**		3.0000	**3.0000**	2.1377e−15	
F19	VPL	− 3.85e+00	Best	− 3.49e+00	− 3.77e+00	9.37e−02	−
	SGO	**− 3.8628**		− 3.8628	**− 3.8628**	2.7101e−15	
F20	VPL	**− 3.32e+00**	Same	− 3.20e+00	− 3.28e+00	5.41e−02	−
	SGO	**− 3.3220**		− 3.2031	**− 3.3061**	0.0411	
F21	VPL	**− 1.02e+01**	Same	− 5.06e+00	**− 9.30e+00**	1.90e+00	+
	SGO	− **10.1532**		− 5.0552	− 8.9012	2.0231	
F22	VPL	**− 1.04e+01**	Same	− 5.09e+00	**− 8.99e+00**	2.35e+00	+
	SGO	**− 10.4029**		− 5.2648	− 8.2761	2.9251	
F23	VPL	**− 1.05e+01**	Same	− 3.57e+00	**− 9.40e+00**	2.28e+00	+
	SGO	**− 10.5364**		− 5.3087	− 7.0231	2.4619	
Total no of the best			02			Total +	03
Total no. of worse			00			Total −	03
Total no. of same			08			Total ≈	04

"−", "+", "≈" denote performance of VPL is worse, better and similar to SGO respectively and same, best and worse are the similar, better and worse solution of SGO then VPL algorithm in term of the best result

such as unimodal, multimodal, separable, and non-separable [132–134]. All benchmark test problems are divided into four categories, such as unimodal separable (US), multimodal separable (MS), unimodal non-separable (UN), and multimodal non-separable (MN). Also, the range, formulation, characteristics, and dimensions of these problems and listed in paper [61].

We have directly derived results of the SELO algorithm from [61], and for results of the SGO algorithm, the codes are implemented in MATLAB 2016a. Experiments are conducted on an Intel-Core-i5, 8 GB memory laptop in Windows 10 environment.

According to parameter settings of the SELO algorithm in its respective paper [61], the common control parameter, such as a maximum number of iterations, is 70,000. In this experiment for each function, the SGO algorithm is tested twice. So, in the first test, we have considered pop_size 50, and the maximum iteration is 250. Hence Max_FEs is ($2 \times 50 \times 250 = 25{,}000$), and in the second test, we have found pop_size is 20, and the maximum iteration is 50, so maximum Max_FEs is ($2 \times 20 \times 50 = 2{,}000$). The other specific parameters for each algorithm are given below.

- SGO setting: For SGO, there is only one parameter C called a self-introspection factor. The value of C is empirically set to 0.2.
- For SELO initial number of families created $M = 03$, number of parents in each family $p = 02$, number of children in each family = 03, parent_follow_probability $r_p = 0.999$, follow_prob_ownparent = 0.999,peer_follow_probability $r_k = 0.1$, follow_prob_factor _otherkids = 0.9991 and sampling interval reduction factor $r = 0.95000$ to 0.99995. Parameters are set as reported by authors in paper [61].

For each benchmark-function, algorithms are run 30 times with different randomly generated populations. Statistical results in terms of mean value, corresponding standard deviation, and best value are reported in Table 21. The table reports the results corresponding to SGO(1) with ($2 \times 50 \times 250$) Max_FEs and results corresponding to SGO(2) with ($2 \times 20 \times 50$) Max_FEs. In this experiment values below 1 E^{-16} are considered to be zero. For every benchmark function, the best results are boldfaced.

To obtain statistically sound conclusions, the WRS test at a 0.05 significance level is conducted on experimental results of Table 21 for both SGO(1) and SGO(2) and reported in the same table. The last three rows of the table summarize results.

Discussion

According Table 21, SGO algorithm achieves success in finding global optimum on F1, F5, F11, F14–F18, F25, F30, F33, F35, F36, F37, F38, F42–F47, F50 functions and near-global optimum for functions F2, F19, F20, F21, F26. For function F3, F4, F6, F12, F13, F14, F19, F21, F22, F27, F28, F29, F36, F37, F48, F49, SGO finds better results than SELO algorithm. For F23, F24, F31, F34, F39, F40, F41 function SELO finds better results than SGO algorithm.

In Table 21, there is '*' mark with the result of the SELO algorithm for the function F14 and F26 to say that there the result might be put wrongly as the minimum value of F14 function is − 1, and minimum value for F26 is − 1.801303410098554. To avoid any conflicts, we have excluded this result and have put '*' in Table 21. From Table 21, according to the WRS test, we find that in 19 cases out of 50 cases SGO(1) algorithm shows better performance than the SELO algorithm and similar to 24 cases out of 50 cases with the SELO algorithm. In contrast, the SELO algorithm shows its better performance in 7 cases out of 50 than the SGO (1) algorithm. Similarly, we find that in 19 cases out of 50 cases SGO (2) algorithm shows better performance than SELO algorithm and similar with 20 cases out of 50 cases with SELO algorithm. In contrast, the SELO algorithm shows its better performance in eleven cases out of 50 than the SGO (2) algorithm.

Hence, we find that the SGO algorithm shows superior performance than the SELO algorithm in this experiment. However, max_FEs for SGO is much less than max_FEs for SELO algorithm, i.e. (25,000 FEs in SGO(1) < 70,000 iterations) and (2000 FEs in SGO(2) < 70,000 iterations). So we can claim that the SGO algorithm outperformed than SELO algorithm in Experiment 5.

Experiment 6

In this experiment, HHO (Harris hawks optimization) [31], SSA (Salp swarm algorithm) [27], GOA (grasshopper optimisation algorithm) [28] and SGO (Social Group optimization) [54] algorithm are compared together. For comparison of the performance of algorithms, a set of 29 benchmark functions are considered. Details of these benchmark functions are given in experiment 1.

We have directly derived results of the HHO algorithm from [31], and the results of the SGO algorithm, SSA algorithm, and GOA algorithm, the codes are implemented in MATLAB 2016a. The source code for the SSA algorithm and GOA algorithm is taken from https://www.alimirjalili.com/SSA.html and https://www.alimirjalili.com/GOA.html, respectively. Experiments are conducted on an Intel-Core−i5, 8 GB memory laptop in Windows 10 environment.

According to parameter settings of the HHO algorithm in its respective paper [31], the common control parameter, such as a maximum number of iterations is 500, and Pop_size is 30. So, for the GOA algorithm and SSA algorithm, the maximum number of iteration and pop_size is set to 500

Table 21 Comparison on SELO and SGO on 30 repetitions

Fun no.	Function name	Algorithms	Mean	WRS test	Std	Best
F1	Foxholes	SELO	**0.9980038538690870**		0.0000013769725269	0.9980038383022720
		SGO(1)	**0.998003837794450**	≈	0	0.998003837794450
		SGO(2)	**0.998003837794450**	≈	0	0.998003837794450
F2	Goldstein-Price	SELO	**3.0013971187248700**		0.0018936009191261	3.0000021202023800
		SGO(1)	**2.999999999999922**	≈	0	2.999999999999922
		SGO(2)	**2.999999999999932**	≈	0	2.999999999999922
F3	Penalized	SELO	0.2899597890213580		0.0159272187796787	0.2497905224307240
		SGO(1)	**6.626245443150890e−08**	−	7.52597822238106e−08	3.058821148321699e−09
		SGO(2)	**0.010200028675921**	−	0.009202450188521	0.001628725501707
F4	Penalizd2	SELO	2.3720510573781100		0.1531241868389090	2.0664368584658500
		SGO(1)	**2.877561733141524e−06**	−	5.85184409308598e−06	7.490617403906734e−08
		SGO(2)	**0.085954598316699**	−	0.054697463554651	0.014253156952674
F5	Ackley	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F6	Beale	SELO	0.0000997928359263		0.0001311815541321	0.0000007530509495
		SGO(1)	**4.368615398517635e−13**	−	5.28892784615697e−13	0
		SGO(2)	**1.895031504568669e−06**	−	2.60630701264704e−06	1.104986488459038e−08
F7	Bonachevsky1	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
		SGO(2)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
F8	Bonachevsky2	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
		SGO(2)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
F9	Bonachevsky3	SELO	**0.0000000000000000**		0.0000000000000001	0.0000000000000000
		SGO(1)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
		SGO(2)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
F10	Booth	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
		SGO(2)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
F11	Branin	SELO	**0.3978943993817670**		0.0003536060523484	0.3978822494361650
		SGO(1)	**0.397887357729738**	≈	0	0.397887357729738
		SGO(2)	**0.397887357747411**	≈	5.5627150160592e−11	0.397887357729738
F12	Colville	SELO	**3.6688019971758100**		1.7577708967227600	0.8388908577815620
		SGO(1)	**1.507864423649453e−04**	≈	2.96207622896498e−04	5.829850050174075e−10
		SGO(2)	**0.882361880633636**	≈	1.387258853940873	1.154167980463728e−04
F13	Dixon-Price	SELO	0.9737369841168760		0.0054869670667257	0.9541730938494050
		SGO(1)	**0.666666670526110**	−	5.3066451465398e−09	0.666666666666667
		SGO(2)	**0.676421575540618**	−	0.003921661696013	0.671115588054493
F14	Easom	SELO	0.0000000000000000*		0.1083854312160620	0.5936514558196160*
		SGO(1)	**− 1**	−	0	− 1
		SGO(2)	**− 1**	−	0	− 1
F15	Fletcher	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
		SGO(2)	2.150797382901120e−06	+	7.05840633570035e−05	3.013791380954771e−09
F16	Fletcher	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000
		SGO(2)	2.467632725158199e−05	+	5.2319588043272e−04	2.980057177690114e−09
F17	Fletcher	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0.0000000000000000**	≈	0.0000000000000000	0.0000000000000000

Table 21 continued

Fun no.	Function name	Algorithms	Mean	WRS test	Std	Best
		SGO(2)	3.080801090891568e−05	+	6.0194808128832e−05	0.274919045293553e−09
F18	Griewank	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F19	Hartman3	SELO	− 2.2922815000937700		0.5795350381767260	− 3.5841184056629400
		SGO(1)	**− 3.862782147820754**	−	0	− 3.862782147820756
		SGO(2)	**− 3.862782147820719**	−	6.5228542918761e−14	− 3.862782147820755
F20	Hartman6	SELO	− 1.1719158908829300		0.0003690446342091	− 1.1727699585993300
		SGO(1)	**− 3.322368011415514**	−	0	− 3.322368011415516
		SGO(2)	**− 3.322364207472817**	−	0.026555990240542	− 3.322367990993992
F21	Kowalik	SELO	0.0003493601571991		0.0000226057336871	0.0003226283751593
		SGO(1)	**3.076983923252769e−04**	−	5.61846636883800e−07	3.074859878057995e−04
		SGO(2)	**3.077095661972494e−04**	−	1.0500275462894e−04	3.076322880214414e−04
F22	Langermann2	SELO	− 1.0835400071766800		0.5277882902242550	− 2.1933014645645000
		SGO(1)	**− 1.811847812568286**	−	0.005275898770057	− 1.816046503083688
		SGO(2)	**− 1.805545479911096**	−	0.043450476166699	− 1.816031587779646
F23	Langermann5	SELO	**− 1.4999998390866700**		0.0000000818646069	− 1.4999999590992100
		SGO(1)	− 1.0966771227191070	+	0.169499381875222	− 1.096711632260348
		SGO(2)	− 1.096643509059383	+	0.252772872708808	− 1.096671357897672
F24	Langermann10	SELO	**− 1.4999991427332700**		0.0000003717669841	− 1.4999999303979900
		SGO(1)	− 0.758576555648690	+	0.298898483760396	− 1.096671262010903
		SGO(2)	− 0.336386278772926	+	0.222342169689975	− 1.096613119161391
F25	Matyas	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F26	Michalewicz2	SELO	− 1.8166465888521900*		0.0072804985619476	− 1.8106292157333700*
		SGO(1)	**− 1.801303410098554**	−	6.83438873551149e−16	− 1.801303410098554
		SGO(2)	**− 1.801303410098554**	−	9.42055475210266e−16	− 1.801303410098554
F27	Michalewicz5	SELO	− 3.3591408962129900		0.2009584117455920	− 3.9631157953194900
		SGO(1)	**− 4.630088626421673**	−	0.065864489928322	− 4.687658179088150
		SGO(2)	**− 4.470327732745007**	−	0.185850086498683	− 4.687327806500385
F28	Michalewicz10	SELO	− 3.9793838974626000		0.0005104314209355	− 3.9806353395021300
		SGO(1)	**− 9.006441111102800**	−	0.311393097467809	− 9.443832461708885
		SGO(2)	**− 9.000011111102800**	−	0.216734109908823	− 9.22441111102800
F29	Perm	SELO	2.0169277899221400		1.2374893392409200	0.3208703882956150
		SGO(1)	**0.118539980853471**	−	0.177190100467076	8.561870933468390e−04
		SGO(2)	**1.561978679351149**	−	1.880078781358827	0.100091999152719
F30	Powell	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F31	Powersum	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	0.006946315911962	+	0.007668016054973	2.606821962241941e−04
		SGO(2)	0.098635720687768	+	0.084418253336401	0.010937757009189
F32	Quartic	SELO	**0.0000989055208389**		0.0000521772789680	0.0000104209894311
		SGO(1)	**2.4890e−05**	≈	1.0695e−05	4.0050e−06
		SGO(2)	**5.600438682972288e−04**	≈	3.86799368671367e−04	7.704746295587896e−05
F33	Rastrigin	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F34	Rosenbrock	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	25.035766221298609	+	0.611250805204646	23.907706666327510

Table 21 continued

Fun no.	Function name	Algorithms	Mean	WRS test	Std	Best
		SGO(2)	28.429285971794421	+	0.148799583491346	28.038832676808092
F35	Schaffer	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	0.002428978248610	+	0.004316407618997	0
F36	Schwefel	SELO	− 0.3402784042291390		3.2212919091274600	− 0.0325083488969540
		SGO(1)	**− 8.461065358958511e+03**	−	1.51802365024271e+03	− 1.114791885433872e+04
		SGO(2)	**− 5.205982653292700e+03**	−	1.06445694628382e+03	− 8.685216618435348e+03
F37	Schwefel 1.2	SELO	0.0000000000000009		0.0000000000000001	0.0000000000000007
		SGO(1)	**0**	−	0	0
		SGO(2)	**0**	−	0	0
F38	Schwefel 2.22	SELO	0.000000000000000		0.000000000000000	0.000000000000000
		SGO(1)	0	≈	0	0
		SGO(2)	0	≈	0	0
F39	Shekel10	SELO	**− 10.536281667618100**		0.0000481237097736	− 10.5363928369535000
		SGO(1)	− 8.128589453290765	+	3.74659045343333e−04	− 10.5363989876543000
		SGO(2)	− 7.893427865009119	+	3.39874357819912	− 9.06579934232111154
F40	Shekel5	SELO	**− 10.1531669871808000**		0.0000172333322304	− 10.1531973132210000
		SGO(1)	− 7.05529877773245551	+	9.03368988843222e−03	− 10.15329067674322995
		SGO(2)	− 5.98564390999007122	+	3.9067544443900010	− 8.95647808884321001
F41	Shekel17	SELO	**− 10.4028748144797000**		0.0000478046191696	− 10.4029869270437000
		SGO(1)	− 8.2648567899765433	+	0.970478900654556	− 10.402989342189900
		SGO(2)	− 6.98877811908877666	+	3.897656789000011	− 9.9009015643335678
F42	Shubert	SELO	− 186.7153981691330000		0.0190762312882078	− 186.7363874875390000
		SGO(1)	**− 1.867309088310239e+02**	−	2.35095908427784e−14	− 1.867309088310239e+02
		SGO(2)	**− 1.867309025071344e+02**	−	2.79033681398514e−05	− 1.867309088310239e+02
F43	Six ump camelback	SELO	− 1.0303924506027700		0.0025133845110030	− 1.0314918740874000
		SGO(1)	**− 1.031628453489878**	−	1.4408156546143e−16	− 1.031628453489878
		SGO(2)	**− 1.031628453489878**	−	1.44081565461429e−16	− 1.031628453489878
F44	Sphere2	SELO	**0.000000000000000**		0.000000000000000	0.000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F45	Step2	SELO	0.0000000000000010		0.0000000000000001	0.0000000000000006
		SGO(1)	**0**	−	0	0
		SGO(2)	**0**	−	0	0
F46	Stepint	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F47	Sumsquares	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
F48	Trid6	SELO	− 46.672022811734100		1.1721159591339900	− 48.933141750726300
		SGO(1)	**− 50.000000000000128**	−	2.3509590842779e−14	− 50.000000000000171
		SGO(2)	**− 49.271240785282032**	−	0.976855025960445	− 49.999991413401290
F49	Trid10	SELO	− 162.571266865506000		− 162.571266865506000	− 162.922114827822000
		SGO(1)	**− 2.099999999933476e+02**	−	1.2523636396815e−08	− 2.100000000000009e+02

Table 21 continued

Fun no.	Function name	Algorithms	Mean	WRS test	Std	Best
		SGO(2)	**– 2.021218477684638e+02**	–	1.27911440793196007	– 2.04763477684638e+02
F50	Zakharov	SELO	**0.0000000000000000**		0.0000000000000000	0.0000000000000000
		SGO(1)	**0**	≈	0	0
		SGO(2)	**0**	≈	0	0
Total + for SGO(1)	07	Total + for SGO(2)	11			
Total − for SGO(1)	19	Total − for SGO(2)	19			
Total ≈ for SGO(1)	24	Total ≈ for SGO(2)	20			

"−", "+", "≈" denote performance of SELO is worse, better and similar to SGO respectively

and 30, respectively. For the SGO algorithm, the maximum number iteration is set to 250.and pop_size is set 30. Hence max_FEs for SGO ($2 \times 30 \times 250$) are the same with other algorithms. The other specific parameters for each algorithm are given below.

- SGO setting: For SGO, there is only one parameter C called a self-introspection factor. The value of C is empirically set 0.2.
- SSA settings: For SSA, there is a parameter $c_1 = 2 \times e^{-(\frac{4}{L})^2}$, where L = max_iteration = 500 as in [27]
- GOA settings: For GOA, $c_{\max}$=1, $c_{\min}$ =0.00004 for finding value of $c = c_{\max} - l \times ((c_{\max} - c_{\min})$/Max_iter), Max_iter = 500 as in [28]
- HHO setting: referred to paper [31].

In this experiment, all algorithms are utilized to tackle scalable unimodal and multimodal F1–F13 test cases with 30, 100, 500, and 1000 dimensions. For each benchmark-function, algorithms are run 30 times with different randomly generated populations. Statistical results in terms of mean value, corresponding standard-deviation are reported in tables. Table 22 reports the result for 30 dimensions. Table 23 reports the result for 100 dimensions. Table 24 reports the result for 500 dimensions. Table 25 reports the result for 1000 dimensions, and Table 26 reports the result for fixed dimensional multimodal and composite benchmark functions. For every benchmark function, the best results are boldfaced.

To obtain statistically sound conclusions, the WRS test at a 0.05 significance level is conducted on experimental results of Tables 22, 23, 24, 25 and 26 and reported in their respective tables. The last three rows of each respective table summarize experimental results.

Discussion

According to Tables 22, 23, 24 and 25, SGO has gained the best performance and consistently performed better than SSA and GOA algorithms. SGO has an excellent performance in exploitation as well as an exploration than SSA and GOA algorithms. It is clear from the results that SGO achieves success in finding global optimum on F1, F3, F9, and F11. HHO algorithm makes success in finding global optimum for the function F9 and F11. For F5, F6, F8, F12, F13, the performance of HHO is better than SSA, GOA, and SGO, whereas, for F1, F2, F3, F4, F7, the performance of SGO is better than HHO, SSA, and GOA. For F19, F10, and F11, both HHO and SGO find equivalent solutions.

From Table 22, according to the WRS test, we find that the SGO algorithm performs superior to the HHO algorithm in five cases, equivalent with four cases out of 13 in 30-dimensional functions. Similarly, the SGO algorithm performs superior to the SSA algorithm in 12 cases and from the GOA algorithm in 13 cases out of 13. In contrast, the HHO algorithm performs superior to the SGO algorithm in four cases, and the SSA algorithm performs superior to SGO in one case. Hence it is seen that the SGO algorithm is outperforming than HHO, SSA, and GOA algorithm in solving 30-dimensional benchmark functions. From Table 23, according to the WRS test, we find that the SGO algorithm performs superior to the HHO algorithm in five cases, equivalent with three cases out of 13 in 100-dimensional functions.

Similarly, the SGO algorithm performs superior to the SSA algorithm in 13 cases and from the GOA algorithm in 13 cases out of 13. In contrast, the HHO algorithm performs superior to the SGO algorithm in five cases. Hence it is seen that both SGO and HHO algorithms are performing equivalent performance, and both SGO and HHO algorithms are outperforming than SSA and GOA algorithm in solving 100-dimensional benchmark functions. From Table 24, according to the WRS test, we find that the SGO algorithm performs superior to the HHO algorithm in five cases, equivalent with three cases out of 13 in 500-dimensional functions. Similarly, the SGO algorithm performs superior to the SSA algorithm in 13 cases and from the GOA algorithm in 13 cases out of 13. In contrast, the HHO algorithm performs superior to the SGO algorithm in five cases. Hence it is seen that the SGO algorithm is performing equivalent performance with the HHO algorithm and outperforming than SSA and GOA algorithm in solving 500-dimensional benchmark functions.

Table 22 Comparison result on HHO, SSA, GOA, and SGO on 3O repetitions on 30 dimension

Functions	HHO		SSA		GOA		SGO
	Results	WRS test	Results	WRS test	Results	WRS test	Results
F1							
Mean	3.95e−97	−	2.6663e−07	−	32.1323	−	**0**
Std	1.72e−96		3.3145e−07		25.0649		**0**
F2							
Mean	1.56e−51	−	2.7193	−	9.6642	−	**4.1502e−173**
Std	6.98e−51		1.9372		4.7732		**0**
F3							
Mean	1.92e−6	−	1.3772e+03	−	2.7763e+03	−	**0**
Std	1.05e−62		663.1158		979.5025		**0**
F4							
Mean	1.02e−47	−	11.3980	−	13.3983	−	**1.5744e−173**
Std	5.01e−47		3.1804		1.8694		**0**
F5							
Mean	**1.32e−02**	+	311.3904	−	4.6817e+03	−	26.4532
Std	**1.87e−02**		469.2823		2.9084e+03		0.5254
F6							
Mean	1.15e−04	≈	**2.1058e−07**	+	26.9521	−	1.5225e−04
Std	1.56e−04		**3.8546e−07**		12.3120		0.0081
F7							
Mean	1.40e−04	−	0.1738	−	0.0405	−	**7.5202e−05**
Std	1.07e−04		0.0768		0.0106		**5.2364e−05**
F8							
Mean	**− 1.25e+04**	+	**−** 7.4013e+03	−	**−** 7.3496e+03	−	**−** 8.4145e+03
Std	**1.47e+02**		639.3972		479.3866		1.0912e+03
F9							
Mean	**0.00e+00**	≈	54.2915	−	81.0099	−	**0**
Std	**0.00e+00**		22.3942		22.1101		**0**
F10							
Mean	**8.88e−16**	≈	2.4587	−	4.6578	−	**8.8818e−16**
Std	4.01e−31		0.8542		0.7395		**0**
F11							
Mean	**0.00e+00**	≈	0.0195	−	1.0406	−	**0**
Std	**0.00e+00**		0.0167		0.1348		**0**
F12							
Mean	**2.08e−06**	+	6.1578	−	7.4999	−	9.3737e−05
Std	**1.19e−05**		2.2550		3.3588		2.1044e−04
F13							
Mean	**1.57e−04**	+	12.0079	−	49.1074	−	3.3322e−04
Std	**2.15e−04**		15.4045		14.0232		2.3423e−03
Total +		04		01		00	
Total −		05		12		13	
Total ≈		04		00		00	

"−", "+", "≈" denote that performance of HHO, SSA and GOA are worse, better and similar to SGO respectively

Table 23 Comparison result on HHO, SSA, GOA and SGO on 3O repetitions on 100 dimension

Functions	HHO		SSA		GOA		SGO
	Results	WRS test	results	WRS test	Results	WRS test	Results
F1							
Mean	1.91e−94	−	1.3120e+03	−	1.1420e+04	−	**0**
Std	8.66e−94		368.2236		1.2257e+03		**0**
F2							
Mean	9.98e−52	−	46.1574	−	91.9275	−	**6.1579e−173**
Std	2.66e−51		7.0802		9.9197		**0**
F3							
Mean	1.84e−59	−	5.4964e+04	−	6.2409e+04	−	**0**
Std	1.01e−58		3.0366e+04		1.0232e+04		**0**
F4							
Mean	8.76e−47	−	28.6121	−	30.7976	−	**1.7327e−173**
Std	4.79e−46		3.1910		3.7461		**0**
F5							
Mean	**2.36e−02**	+	1.5948e+05	−	3.9104e+06	−	96.7537
Std	**2.99e−02**		7.3288e+04		1.1528e+06		0.4560
F6							
Mean	**5.12e−04**	+	1.4014e+03	−	1.1686e+04	−	0.0285
Std	**6.77e−04**		421.3000		1.5398e+03		0.1388
F7							
Mean	1.85e−04	−	2.6735	−	0.6814	−	**7.1811e−05**
Std	4.06e−04		0.5435		0.1922		**5.0044e−05**
F8							
Mean	**− 4.19e+04**	+	**−** 2.1568e+04	−	**−** 1.7308e+04	−	**−** 2.4282e+04
Std	**2.82e+00**		1.5894e+03		2.4081e+03		4.7056e+03
F9							
Mean	**0.00e+00**	≈	232.0861	−	404.9512	−	**0**
Std	**0.00e+00**		43.8589		39.2338		**0**
F10							
Mean	**8.88e−16**	≈	9.9830	−	13.3441	−	**8.8818e−16**
Std	**4.01e−31**		1.0140		0.7088		**0**
F11							
Mean	**0.00e+00**	≈	15.0274	−	101.8061	−	**0**
Std	**0.00e+00**		4.5248		10.9556		**0**
F12							
Mean	**4.23e−06**	+	33.1875	−	7.6639e+03	−	2.2716e−04
Std	**5.25e−06**		11.1185		8.7399e+03		0.0018
F13							
Mean	**9.13e−05**	+	7.9763e+03	−	2.6647e+06	−	0.0230
Std	**1.26e−04**		1.3000e+04		2.1350e+06		1.7442
Total +		05		00		00	
Total −		05		13		13	
Total ≈		03		00		00	

"−", "+", "≈" denote that performance of HHO, SSA and GOA are worse, better and similar to SGO respectively

Table 24 Comparison result on HHO, SSA, GOA and SGO on 3O repetitions on 500 dimension

Functions	HHO		SSA		GOA		SGO
	Results	WRS test	Results	WRS test	Results	WRS test	Results
F1							
Mean	1.46e−92	−	9.4569e+04	−	8.7478e+04	−	**0**
Std	8.01e−92		6.2059e+03		4.3559e+03		**0**
F2							
Mean	7.87e−49	−	530.3178	−	500.1891	−	**4.2194e−172**
Std	3.11e−48		19.8194		10.6519		**0**
F3							
Mean	6.54e−37	−	1.4517e+06	−	4.5669e+14	−	**0**
Std	3.58e−36		6.2641e+05		2.9900e+14		**0**
F4							
Mean	1.29e−47	−	40.1866	−	32.9244	−	**1.7939e−173**
Std	4.11e−47		2.7635		41.9023		**0**
F5							
Mean	**3.10e−01**	+	3.6217e+07	−	7.6774e+07	−	493.9169
Std	**3.73e−01**		4.2649e+06		6.9429e+06		0.3720
F6							
Mean	**2.94e−03**	+	9.3480e+04	−	1.3352e+05	−	4.6227
Std	**3.98e−03**		7.0249e+03		8.7871e+03		2.4453
F7							
Mean	2.51e−04	−	276.0543	−	117.6845	−	**9.3814e−05**
Std	2.43e−04		45.7197		267.8912		**5.8056e−05**
F8							
Mean	**− 2.09e+05**	+	**−** 5.9853e+04	−	**−** 4.7092e+04	−	**−** 8.2593e+04
Std	**2.84e+01**		4.2282e+03		2.9012e+04		2.2958e+04
F9							
Mean	**0.00e+00**	≈	3.1838e+03	−	4.8912e+03	−	**0**
Std	**0.00e+00**		119.4264		2.7615e+03		**0**
F10							
Mean	**8.88e−16**	≈	14.2634	−	891.2316	−	**8.8818e−16**
Std	**4.01e−31**		0.2265		262.5149		**0**
F11							
Mean	**0.00e+00**	≈	854.4929	−	13.0011	−	**0**
Std	**0.00e+00**		66.3466		0.9623		**0**
F12							
Mean	**1.41e−06**	+	1.3346e+06	−	1.8308e+07	−	8.5429e−04
Std	**1.48e−06**		8.0189e+05		4.5409e+06		0.0113
F13							
Mean	**3.44e−04**	+	3.6253e+07	−	9.6599e+07		0.3790
Std	**4.75e−04**		9.4885e+06		2.9012e+06		19.2315
Total +		05		00		00	
Total −		05		13		13	
Total ≈		03		00		00	

"−", "+" and "≈" denote that performance of HHO, SSA, and GOA are worse, better and similar to SGO respectively

Table 25 Comparison result on HHO, SSA, GOA, and SGO on 3O repetitions on 1000 dimension

Functions	HHO		SSA		GOA		SGO
	Results	WRS test	Results	WRS test	Results	WRS test	Results
F1							
Mean	1.06e−94	−	2.3895e+05	−	4.6729e+05	−	**0**
Std	4.97e−94		1.0130e+04		3.9023e+04		**0**
F2							
Mean	2.52e−50	−	1.1980e+03	−	978.7342	−	**8.8515e−172**
Std	5.02e−50		25.9659		23.3419		**0**
F3							
Mean	1.79e−17	−	5.5065e+06	−	4.8911e+16	−	**0**
Std	9.81e−17		2.1001e+06		2.0211e+16		**0**
F4							
Mean	1.43e−46	−	46.0339	−	61.5621	−	**1.8015e−173**
Std	7.74e−46		2.2454		10.9793		**0**
F5							
Mean	**5.73e−01**	+	1.2061e+08	−	2.0374e+08	−	989.8076
Std	**1.40e+00**		1.0872e+07		1.8934e+07		0.9994
F6							
Mean	**3.61e−03**	+	2.2885e+05	−	2.9168e+05	−	10.8339
Std	**5.38e−03**		9.0680e+03		5.8715e+03		6.4262
F7							
Mean	1.41e−04	−	1.7908e+03	−	680.8912	−	**7.5620e−05**
Std	1.63e−04		178.7950		92.1475		**4.7538e−05**
F8							
Mean	**− 4.19e+05**	+	**−** 8.7049e+04	−	**−** 6.4190e+04	−	**−** 1.5102e+05
Std	**1.03e+02**		9.3834e+03		3.5412e+03		4.7114e+04
F9							
Mean	**0.00e+00**	≈	7.5972e+03	−	7.0342e+03	−	**0**
Std	**0.00e+00**		205.0048		3.7534e+03		**0**
F10							
Mean	**8.88e−16**	≈	14.5380	−	3.6790e+03	−	**8.8818e−16**
Std	**4.01e−31**		0.1588		1.1832e+03		**0**
F11							
Mean	**0.00e+00**	≈	2.1056e+03	−	13.9612	−	**0**
Std	**0.00e+00**		102.3777		0.4812		**0**
F12							
Mean	**1.02e−06**	+	1.1915e+07	−	3.8182e+07	−	0.0022
Std	**1.16e−06**		3.2836e+06		1.9756e+07		0.0078
F13							
Mean	**8.41e−04**	+	1.4035e+08	−	3.5707e+08	−	2.0727
Std	**8.41e−04**		1.9003e+07		6.4523e+07		42.0966
Total +		05		00		00	
Total −		05		13		13	
Total ≈		03		00		00	

"−", "+" and "≈" denote that performance of HHO, SSA, and GOA are worse, better and similar to SGO respectively

Table 26 Comparison result on HHO, SSA, GOA and SGO on 3O repetitions on fixed-dimensional and composite benchmark functions

Functions	HHO		SSA		GOA		SGO
	Results	WRS test	Results	WRS test	Results	WRS test	Results
F14							
Mean	**9.98e−01**	≈	1.0643	−	**0.9980**	≈	**0.9980**
Std	**9.23e−01**		0.2567		**1.5701e−16**		**3.4303e−08**
F15							
Mean	**3.10e−04**	≈	**9.1497e−04**	≈	0.0086	−	**3.1049e−04**
Std	**1.97e−04**		**2.8876e−04**		0.0101		**2.3310e−05**
F16							
Mean	**− 1.03e+00**	≈	**− 1.0316**	≈	**− 1.0316**	≈	**− 1.0316**
Std	**6.78e−16**		**2.4267e−14**		**1.5289e−13**		**6.7752e−16**
F17							
Mean	**3.98e−01**	≈	**0.3979**	≈	**0.3979**	≈	**0.3979**
Std	**2.54e−06**		**2.6978e−04**		**3.8926e−13**		**4.5168e−16**
F18							
Mean	**3.00e+00**	≈	**3.0000**	≈	**3.0000**	≈	**3.0000**
Std	**0.00e+00**		**2.2603e−13**		**3.2446e−12**		**1.5003e−15**
F19							
Mean	**− 3.86e+00**	≈	**− 3.8628**	≈	**− 3.8589**	−	**− 3.8628**
Std	**2.44e−03**		**2.3584e−11**		**0.0084**		**2.2584e−16**
F20							
Mean	**− 3.322**	≈	− 3.2222	−	− 3.2729	−	**− 3.3220**
Std	**0.137406**		0.0521		0.0673		**0.0363**
F21							
Mean	**− 10.1451**	+	− 6.8073	−	− 6.1441	−	− 7.0552
Std	**0.885673**		3.3663		3.7934		9.0336e−03
F22							
Mean	**− 10.4015**	+	− 7.5677	−	− 6.2905	−	− 8.2648
Std	**1.352375**		3.6407		3.8768		0.9704
F23							
Mean	**− 10.5364**	+	− 8.4837	−	− 4.2256	−	− 8.1285
Std	**0.927655**		3.5235		3.5349		3.7465
F24							
Mean	396.8256	−	**20.0000**	+	180.0000	−	56.6667
Std	79.58214		**44.7214**		148.3240		67.8911
F25							
Mean	910	−	**35.0843**	+	378.5491	−	99.9197
Std	0		**43.9729**		113.3010		90.4498
F26							
Mean	910	−	**202.1367**	+	342.4062	−	228.5903
Std	0		**83.0302**		157.5014		46.0290
F27							
Mean	910	−	320.4287	+	**286.0073**	+	457.9925
Std	0		33.0710		**139.0947**		110.2783
F28							
Mean	860.8925	−	**25.2016**	+	199.6790	−	59.3325
Std	0.651222		**41.8286**		120.4731		70.9304

Table 26 continued

Functions	HHO		SSA		GOA		SGO
	Results	WRS test	Results	WRS test	Results	WRS test	Results
F29							
Mean	558.9653	−	582.7280	−	**206.6856**	+	486.6667
Std	5.112352		179.4161		**90.5928**		73.0297
Total +		03		05		02	
Total −		06		06		10	
Total ≈		07		05		04	

"−", "+" and "≈" denote that performance of HHO, SSA, and GOA are worse, better and similar to SGO respectively

From Table 25, according to Wilcoxon's rank-sum test, we find that the SGO algorithm performs superior to the HHO algorithm in five cases, equivalent with three cases out of 13 in 1000 dimensional functions. Similarly, the SGO algorithm performs superior to the SSA algorithm in 13 cases and from the GOA algorithm in 13 cases out of 13, whereas the HHO algorithm performs superior to the SGO algorithm in five cases. It is seen that the SGO algorithm is performing equivalent performance with the HHO algorithm and outperforming than SSA and GOA algorithm in solving 1000-dimensional benchmark functions.

The fixed-dimensional multimodal functions are designed to have many local optimal where computation complexity increases drastically with the problem size. The results reported from Table 26, it is clear that the SGO algorithm achieves success in finding an optimal solution for functions F14, F16, F17, F18, F19, and F20. Similarly, the GOA algorithm makes success in finding an optimal solution for functions F14, F16, F17, and F18. SSA algorithm finds success for function F16–F19. HHO algorithm finds success for function F14, F16–F20, F22–F23. For the shekel family, the HHO algorithm finds superior solutions than SSA, GOA, and SGO algorithm.

The composite functions are well enough to judge the ability to escape from local minima of meta-heuristics optimization algorithms. The results from Table 26, it is clear that the SSA algorithm finds a superior solution for F24, F25, F26, and F28, and for F27 and F29, GOA finds superior solution than other algorithms. The performance of the HHO algorithm for solving composite benchmark functions is worse than other algorithms.

From Table 26, according to the WRS test, we find that the SGO algorithm performs superior to the HHO algorithm in six cases, equivalent with seven cases out of 16 benchmark functions. Similarly, the SGO algorithm performs superior to the SSA algorithm in six cases, equivalent with five cases out of 16 benchmark functions. SGO algorithm performs superior to the GOA algorithm in ten cases, equivalent with four cases out of 16 benchmark functions. Whereas the HHO algorithm performs superior to the SGO algorithm in three cases, the SSA algorithm performs superior to SGO in five cases, and the GOA algorithm performs superior to SGO in two cases. Hence it is seen that the SGO algorithm is outperforming performance than HHO, SSA and GOA algorithm in solving ten fixed dimensional multimodal and six composite benchmark functions.

Experiment 7: on classical engineering problem

In this experiment, we have applied all the algorithms such as LAPO, GROM, BOA, SSA, VPL, HHO, SELO, SSOA, GOA, and SGO algorithm for solving classical engineering problems. Here we have considered six classical engineering problems. These are.

Tension/compression spring design problem

The objective of this test problem is to minimize the weight of tension/compression spring shown in Fig. 2 [135, 136]. The optimum design must satisfy constraints on shear stress, surge frequency, and deflection. There are three design variables: wire diameter (d), mean coil diameter (D), and many active coils (N). The formulated optimization problem is given in Appendix D.

The optimization result of all algorithms for the compression spring design problem is given in Table 27. The result of LAPO algorithm is reported from paper [79], for GROM the result is reported from [80], for BOA the result is reported from [29], for VPL the result is reported from [60], for HHO the result is reported from [31], for SSA the result is taken reported from [27] and result for SGO is found by us. "NA" stands for an experiment that is not conducted for that algorithm. The best result is represented in boldface. From the table, we find that the BOA algorithm outperforms than all other algorithms.

The welded beam design problem

The objective of this test problem is to minimize the fabrication cost of the welded beam shown in Fig. 3 [137].

Table 27 Comparison of tension/compression spring design problem

Algorithm	Optimum variables			Optimum weight
	d	*D*	*N*	
SGO	0.05167575	0.3563976	11.3077598	0.0126652
LAPO	0.0519038638	0.361890902	11.28885	0.01265722
GROM	0.0517271517	0.357630345	11.2361437	0.01265809
BOA	**0.051343**	**0.334871**	**12.922700**	**0.0119656**
SSOA	NA			
VPL	0.0501910!	0.331680!	12.834269!	0.0123947!
HHO	0.051796393	0.359305355	11.138859	0.012665443
SELO	NA			
SSA	0.051207	0.345215	12.004032	0.0126763
GOA	NA			

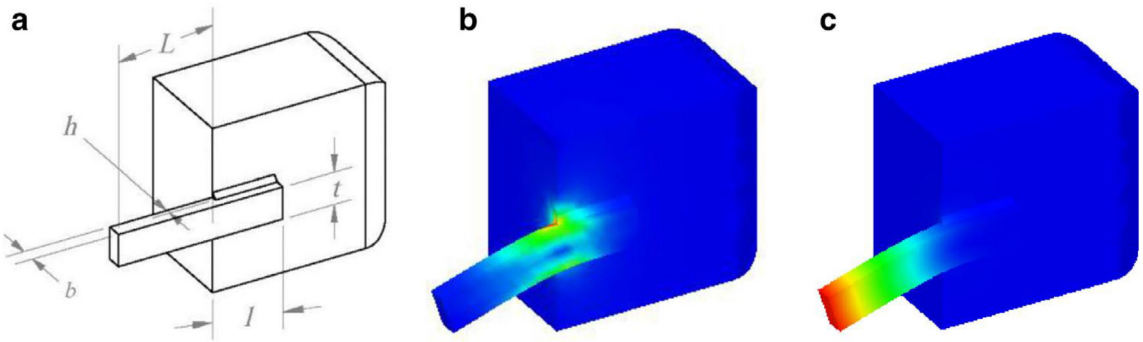

Fig. 3 Welded beam design problem: **a** schematic of the weld; **b** stress distribution evaluated at the optimum design; **c** displacement distribution at the optimum design

Optimization constraints are on shear stress (τ), and bending stress in the beam (θ), buckling load (Pc), end deflection of beam (δ). There are four optimization variables: the thickness of the weld (h), length of clamped bar (l), the height of the bar (t), and thickness of the bar (b). The formulated optimization problem is given in Appendix D.

The optimization result of all algorithms for Welded beam design problem is given in Table 28. The result of LAPO algorithm is reported from paper [79], for GROM the result is reported from [80], for BOA the result is reported from [29], for VPL the result is reported from [60], for HHO the result is reported from [31], for SSA the result is reported from [27]. The result for SGO is found by us. "NA" stands for an experiment that is not conducted for that algorithm. The best result is represented in boldface. From the table, we find that the SGO algorithm outperforms than all other algorithms.

Pressure vessel design problem

This problem goal is to minimize the total cost (material, forming, and welding) of cylindrical pressure vessels shown in Fig. 4 [25]. Both ends of the vessel are capped while the head has a hemispherical shape. There are four optimization variables: the thickness of the shell (T_s), the thickness of the head (T_h), inner radius (R), length of the cylindrical section without considering head (L). The formulated optimization problem is given in Appendix D.

The optimization result of algorithms for the pressure vessel design problem is given in Table 29. The result of the LAPO algorithm is reported from paper [79], for the GROM the result is reported from [80], for the VPL the result is reported from [60], for HHO the result is reported from [31] and result for SGO is found by us. A "NA" stand for the experiment is not conducted for that algorithm. The best result is represented in boldface. From table, we find that the SGO algorithm outperforms all other algorithms.

Cantilever beam design problem

In this problem, the goal is to minimize the weight of a cantilever beam with hollow square blocks. There are five squares of which the first block is fixed, and the fifth one burdens a vertical load, box girders, and lengths of those girders are design parameters for this problem. This cantilever beam design is shown in Fig. 5 [78]. The formulated optimization problem is defined in Appendix D.

Table 28 Comparison of Welded beam design problem

Algorithm	Variables				Optimum cost
	h	*l*	*t*	*b*	
SGO	**0.21542798**	**0.21542796**	**7.41442256**	**0.21542797**	**1.5921491**
LAPO	0.205528028615	3.394773587424	9.076635213428	0.2055309791245	1.71960676580
GROM	0.205530838237860	3.39469488081047	9.07663928640037	0.20553083824796	1.7196019906343
BOA	0.1736	2.9690	8.7637	0.2188	1.6644
SSOA	NA				
VPL	0.215235!	6.898945!	8.815033!	0.216253!	2.264711!
HHO	0.204039	3.531061	9.027463	0.206147	1.73199057
SELO	NA				
SSA	0.2057	3.4714	9.0366	0.2057	1.72491
GOA	NA				

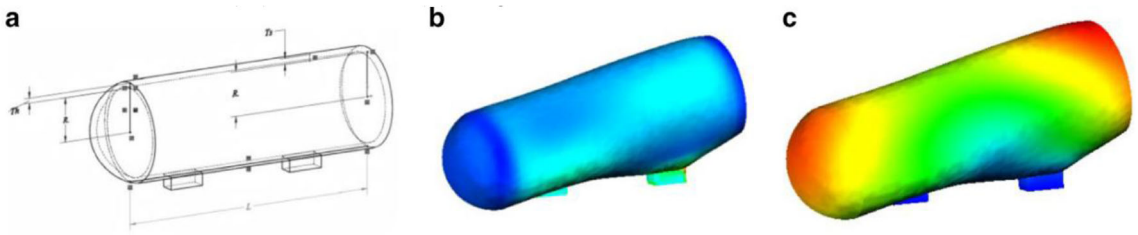

Fig. 4 Pressure vessel design problem: **a** schematic of the vessel; **b** stress distribution evaluated at the optimum design; and **c** displacement distribution evaluated at the optimum design

Table 29 Comparison of pressure vessel design problem

Algorithm	Optimum variables				Optimum cost
	T_s	T_h	*R*	*L*	
SGO	**1.258846989**	**0**	**65.22523267**	**10.00000000**	**2611.921927**
LAPO	0.786283665	0.39160673845	40.758736322	194.296457539	5916.1935786
GROM	0.778168641372626	0.384649162633450	40.3196187241064	200	5885.33277364205
BOA	NA				
SSOA	NA				
VPL	0.815200	0.426500	42.0912541	176.742314	6044.9565
HHO	0.81758383	0.4072927	42.09174576	176.7196352	6000.46259
SELO	NA				
SSA	NA				
GOA	NA				

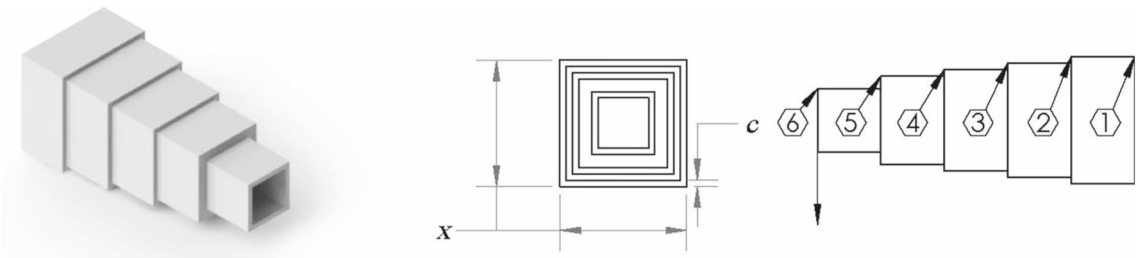

Fig.5 Cantilever beam design

Table 30 Results for cantilever beam design problem

Algorithm	Optimal values for variables					Optimum weight
	x_1	x_2	x_3	x_4	x_5	
SGO	**6.01605811183**	**5.30964910182**	**4.4941343030**	**3.50169590256**	**2.152122757**	**1.3365206094**
LAPO	6.01243634	5.314870556	4.4959135494	3.4993942765	2.151154796	1.336521415
GROM	6.015401113310	5.30998470907	4.49536712598	3.50063527673	2.152272871848	1.33652066668
BOA	NA					
SSOA	NA					
VPL	NA					
HHO	NA					
SELO	NA					
SSA	6.015134526133	5.30930467605	4.4950067163	3.5014262863	2.15278790	1.3399563910
GOA	6.011674	5.31297	4.48307	. 3.50279	2.16333	1.33996

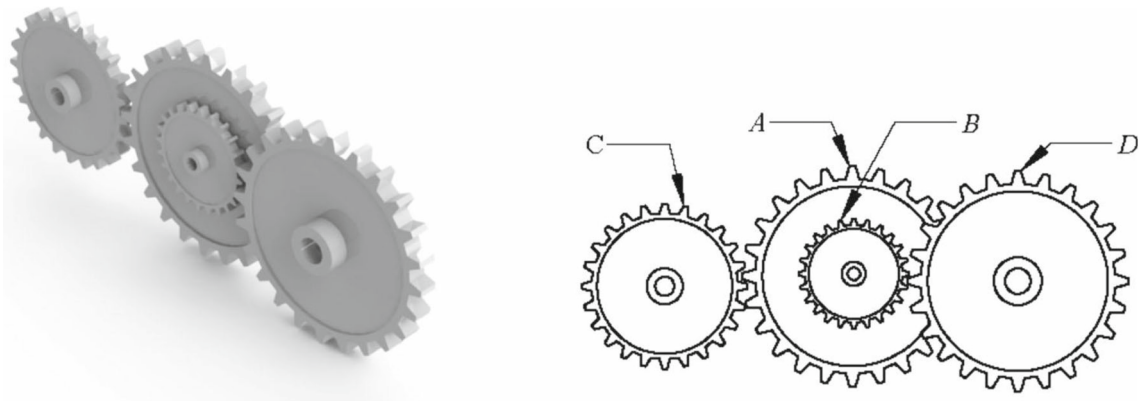

Fig. 6 Gear train design

The optimization result of algorithms for cantilever beam design problem is given in Table 30. The result of LAPO algorithm is reported from paper [79], for GROM the result is reported from [80], for VPL the result is taken from [60], for SSA the result is reported from [27], for GOA the result is reported from [28] and result for SGO is found by us. "NA" stands for the experiment are not conducted for that particular algorithm. The best result is represented in boldface. From the Table, we get that the SGO algorithm outperforms than all other algorithms.

Gear train design problem

The objective of Gear train design problem is to minimize gear ratio where gear ratio is calculated as by Eq. 1:

$$\text{Gear ratio} = \frac{\text{angular velocity of output shaft}}{\text{angular velocity of input shaft}}. \quad (1)$$

This problem has four parameters. The parameters of this problem are discrete with an increment size of 1 since they define teeth of the gears (n_A, n_B, n_C, n_D). These constraints are only limited to the ranges of the variables. This Gear train design is shown in Fig. 6 [78]. The formulated optimization problem is defined in Appendix D.

The optimization result of algorithms for the gear train design problem is given in Table 31. The result of the LAPO algorithm is reported from paper [79]. For GROM, the result is reported from [80], for BOA, the result is reported from [28], the result for SGO is found by us. "NA" stands for the experiment is not conducted for that particular algorithm. The best result is represented by boldface. From the table, we get that all the algorithms perform equally.

Three-bar truss design problem

Here the objective is to design a truss with a minimum weight that does not violate constraints. A most important issue in designing truss is constraints that include stress, deflection, and buckling constraints. Figure 7 [78] shows the structural parameters of this problem. The formulated design problem is given in Appendix D. We can see that the objective function is quite simple, but it is subject to several challenging constraints. This truss design problems are prevalent in the literature of meta-heuristics [138, 139]

Table 31 Comparison results for gear train design problem

Algorithm	Variables				Optimum gear
	n_A	n_B	n_C	n_D	
SGO	**43**	**16**	**19**	**49**	**2.7008571488865e−12**
LAPO	**49**	**16**	**19**	**43**	**2.700857E−12**
GROM	**43**	**16**	**19**	**49**	**2.7008571E−12**
BOA	**43**	**16**	**19**	**49**	**2.7008571E−12**
SSOA	NA				
VPL	NA				
HHO	NA				
SELO	NA				
SSA	NA				
GOA	NA				

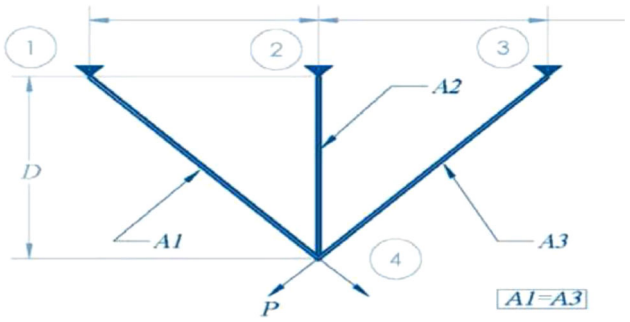

Fig. 7 Three-bar-truss design

The optimization result of algorithms for the three-bar-truss design problem is given in Table 32. The result for the HHO algorithm is reported from paper [31], for SSA, the result is reported from [27], for GOA, the result is reported from [28], the result for SGO is found by us. "NA" stands for the experiment are not conducted for that algorithm. The best result is represented in boldface. From the Table, we get that the SGO algorithm outperforms than all other algorithms.

Overall conclusion

This section applies all the algorithms such as LAPO, GROM, BOA, SSA, VPL, HHO, SELO, SSOA, GOA, and SGO algorithm for solving benchmark functions as well as classical engineering problems. From the above experiments, we conclude that the performance of the SGO algorithm is worse than other algorithms while solving Rosenbrock's benchmark function, shekel family benchmark function, i.e., shekel 2, shekel 5, and shekel 7. While solving composite benchmark functions, SSA (Salp swarm algorithm) is superior, and the HHO algorithm is worse than other algorithms. On solving high dimensional and classical engineering problems, the SGO algorithm is superior to other algorithms.

Conclusion

As we know, meta-heuristic optimization algorithms are more popular than deterministic search optimization algorithms in solving global optimization problems, and several optimization algorithms are proposed to solve global optimization problems. The exploration search and exploitation search are two important factors related to meta-heuristic optimization methods. These two factors are in contrast with each other. In other words, focusing too much on local search, i.e., exploitation may result in getting stuck in local optimum points, and too much focusing on global search, i.e., exploration, may cause the low quality of the final best answer. So an algorithm should be in the form that it can balance in between and find out an optimal solution to the problem. Free-Lunch theorem for optimization says that none of the optimization algorithms can solve all optimization problems and *makes this area of research open. So, researchers improve/adapt the current algorithms for solving different problems or propose new algorithms for providing competitive results compared to the existing algorithms.* As a result,

Table 32 Results of the three-bar truss design problem

Algorithm	Variables		Optimal weight
	x_1	x_2	
SGO	**0.788674579690751**	**0.408249860015748**	**263.8958433810869**
LAPO	NA		
GROM	**NA**		
BOA	NA		
SSOA	NA		
VPL	NA		
HHO	0.788662816	0.408283133832900	263.8958434
SELO			
SSA	0.788665414258065	0.408275784444547	263.8958434
GOA	0.788897555578973	0.407619570115153	263.895881496069

a number of optimization algorithms have been proposed from a few decades to till date. It is difficult to compare all algorithms or challenging to say, which is the best because none of the algorithms can solve all optimization problems. In this paper, we have considered ten algorithms SGO, LAPO, GROM, BOA, SSOA, VPL, HHO, SELO, GOA, and SSA those are proposed in the year 2017–2019 except SGO as SGO is proposed in the year of 2016, and have conducted seven experiments by considering a different type of problems such as mathematical classical optimization problems, CEC global optimization problems, and six classical engineering design problems. We have seen that the SGO algorithm has shown its superior performance in comparison to all other algorithms in solving problems in each experiment. Still, while addressing Rosenbrock benchmark function and shekel family benchmark function, i.e., shekel 2, shekel 5, and shekel 7, the performance of SGO is not so well in comparison to other algorithms. One of the best things we see in SGO is that the performance does not deteriorate as the dimension of the problem increases, and at the same time, it can find an optimum result in less fitness evaluation. As further research, we improve the performance of the SGO algorithm by balancing between exploration and exploitation search so that it can find the optimal solution of benchmark function like shekel family and other. Also, we compare SGO performance with newly proposed optimization algorithms, solving the problem of large-scale optimizations and multi-objective optimizations.

Compliance with ethical standards

Conflict of interest Authors declares that they have no conflict of interest on publication of this paper

Appendix A

See Tables 33, 34, 35, and 36.

Table 33 Unimodal benchmark functions

Function	Dim	Range	$f_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[− 100, 100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i-1}^{n} \lvert x_i \rvert$	30	[− 10, 10]	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	30	[− 100,1 00]	0
$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	[− 100, 100]	0
$F_5(x) = \sum_{i=1}^{n-1} [100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2]$	30	[− 30, 30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[− 100, 100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	30	[− 1.28, 1.28]	0

Table 34 Multi-modal benchmark functions

Function	Dim	Range	$f_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	30	[− 500, 500]	0
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[− 5.12, 5.12]	0
$F_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i))+20 + e$	30	[− 32, 32]	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[− 600, 600]	0
$F_{12}(x) =$	30	[− 50, 50]	0
$F_{12}(x) = \frac{\pi}{n}\{10\sin(\pi y_i)+\sum_{i=1}^{n-1} (y_i - 1)^2\left[1 + 10sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i - a)^m x_i < -a \end{cases}$	30	[− 50, 50]	0
$F_{13}(x)$=0.1$\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2[1 + \sin^2(3\pi x_i+1)] + (x_n - 1)^2[1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i,$ 5, 100, 4)			

Table 35 Fixed –dimensional multimodal benchmark functions

Function	Dim	Range	$f_{\min}$
$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j+\sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	− [65, 65]	1
$F_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}]^2$	4	[− 5, 5]	0.00030
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[− 5, 5]	− 1.0316
$F_{17}(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	2	[− 5, 5]	0.398
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2+6x_1x_2+3x_2^2)]$ $X[30+(2x_1 - 3x_2)^2 x(18–32x_1+12x_1^2+48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[− 2, 2]	3
$F_{19}(x) = -\sum_{i=1}^{4} c_1 \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	[1, 3]	− 3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_1 \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	[0, 1]	− 3.32
$F_{21}(x) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0. 10]	− 10.1532
$F_{22}(x) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0. 10]	− 10.4028
$F_{23}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0. 10]	− 10.5363

Table 36 Composite functions

Functions	Dim	Range	f_{min}
$F_{24}(CF1)$ $f_1, f_2, f_3, \ldots, f_{10}$ = Sphere function $[\sigma_1, \sigma_2, \sigma_3, \ldots\ldots, \sigma_{10}] = [1,1,1,\ldots,11]$ $[\lambda_1, \lambda_2, \lambda_3 \ldots\ldots, \lambda_{10}] = [\frac{5}{100}, \frac{5}{100}, \frac{5}{100}, \ldots, \frac{5}{100}]$	30	[− 5, 5]	0
$F_{25}(CF2)$ $f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's function $[\sigma_1, \sigma_2, \sigma_3, \ldots\ldots, \sigma_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3 \ldots\ldots, \lambda_{10}] = [\frac{5}{100}, \frac{5}{100}, \frac{5}{100}, \ldots, \frac{5}{100}]$	30	[− 5, 5]	0
$F_{26}(CF3)$ $f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's Function $[\sigma_1, \sigma_2, \sigma_3, \ldots\ldots, \sigma_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3 \ldots\ldots, \lambda_{10}] = [1,1,1,\ldots,1]$	30	[− 5, 5]	0
$F_{27}(CF4)$ f_1, f_2 = Ackley's function, f_3, f_4 = Rastrgin's function, f_5, f_6 = Weierstrass function, f_7, f_8 = Griewank's function, f_9, f_{10} = Sphere function $[\sigma_1, \sigma_2, \sigma_3, \ldots\ldots, \sigma_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3 \ldots\ldots, \lambda_{10}] = [\frac{5}{32}, \frac{5}{32}, 1, 1, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}, \frac{5}{100}]$	30	[− 5, 5]	0
$F_{28}(CF5)$ f_1, f_2 = Rastrigin's function, f_3, f_4 = Weierstrass function, f_5, f_6 = Griewank's function, f_7, f_8 = Ackley's function, f_9, f_{10} = Sphere function $[\sigma_1, \sigma_2, \sigma_3, \ldots\ldots, \sigma_{10}] = [1,1,1,\ldots,1]$ $[\lambda_1, \lambda_2, \lambda_3 \ldots\ldots, \lambda_{10}] = [\frac{1}{5}, \frac{1}{5}, \frac{5}{0.5}, \frac{5}{0.5}, \frac{5}{100}, \frac{5}{100}, \frac{5}{32}, \frac{5}{32}, \frac{5}{100}, \frac{5}{100}]$	30	[− 5, 5]	0
$F_{29}(CF6)$ f_1, f_2 = Rastrigin's function, f_3, f_4 = Weierstrass function, f_5, f_6 = Griewank's function, f_7, f_8 = Ackley's function, f_9, f_{10} = sphere function $[\sigma_1, \sigma_2, \sigma_3, \ldots\ldots, \sigma_{10}] = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]$ $[\lambda_1, \lambda_2, \lambda_3 \ldots\ldots, \lambda_{10}] = [0.1 \times \frac{1}{5}, 0.2 \times \frac{1}{5}, 0.3 \times \frac{5}{0.5}, 0.4 \times \frac{5}{0.5}, 0.5 \times \frac{5}{100}, 0.6 \times \frac{5}{100}, 0.7 \times \frac{5}{32}, 0.8 \times \frac{5}{32}, 0.9 \times \frac{5}{100}, 1 \times \frac{5}{100}]$	30	[− 5, 5]	0

Appendix B

See Tables 37.

Table 37 Benchmark functions used in butterfly optimization algorithm

No	Benchmark functions	Type	Formula	Dim	range	$f_{\min}$
F1	Sphere	M.S	$f(x) = \sum_{i=1}^{D} x_i^2$	30	[− 100, 100]	0
F2	Beale	U.N	$f(x) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	2	[− 4.5, 4.5]	0
F3	Cigar	U.N	$f(x) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$	30	[− 100, 100]	0
F4	Step	U.S	$f(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	30	[− 100, 100]	0
F5	Quartic function with no ise	U.S	$f(x) = \sum_{i=1}^{D} ix_i^4 + \text{random}(0, 1)$	30	[− 1.28, 1.28]	0
F6	Bohachevsky	M.N	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	[− 100, 100]	0
F7	Ackley	M.N	$(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - 20 + e$	30	[− 32, 32]	0
F8	Griewank	M.N	$f(x) = \frac{1}{4000}\sum_{i=\exp\left(\frac{1}{n}\sum_{i=1}^{D}\cos\left(2\times pi\times x_i\right)\right)+1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[− 600, 600]	0
F9	Levy	M.S	$f(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2\left[1 + \sin^2(3x_2)\right] + (x_1 - 1)^2\left[1 + \sin^2(2\pi x_2)\right]$	2	[− 10, 10]	0
F10	Michalewiz	M.S	$f(x) = -\sum_{i=1}^{D}\sin(x_i)(\sin(ix_i^2/\pi))^{2m}, m = 10$	10	[0, pi]	− 0.966015
F11	Rastrigin	M.S	$f(x) = \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[− 5.12 ,5.12]	
F12	Alpine	M.S	$f(x) = \sum_{i=1}^{D}\lvert x_i\sin\left(x_i\right) + 0.1x_i\rvert$	30	[− 10, 10]	0
F13	Schaffer	M.N	$f(x) = \frac{sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	2	[− 100, 100]	0
F14	Rosenbrock	U.N	$f(x) = \sum_{i=1}^{D-1}[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2]$	30	[− 10,10]	0
F15	Easom	M.S	$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	2	[− 100, 100]	− 1
F16	Shubert	M.S	$f(x) = (\sum_{i=1}^{5} i\cos((i + 1)x_1 + i))(\sum_{i=1}^{5} i\cos((i + 1)x_2 + i))$	2	[− 10, 10]	− 186.7309
F17	Schwefel 1.2	U.N	$f(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	30	[− 10, 10]	0
F18	Schwefel 2.21	U.S	$f(x) = \max_i\{\lvert x_i\rvert, 1 \le i \le D\}$	30	[− 10, 10]	0
F19	Schwefel 2.22	U.N	$f(x) = \sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	30	[− 10, 10]	0
F20	Schwefel 2.26	M.S	$f(x) = \sum_{i=1}^{D}[x_i\sin(\sqrt{\lvert x_i\rvert})]$	30	[− 500, 500]	− 418.982XD
F21	Booth	U.N	$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	2	[− 10, 10]	0
F22	Goldstein price	M.N	$= [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$ $X[30 + (2x_1 - 3x_2)^2 x(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[− 2, 2]	3
F23	Matyas	U.N	$f(x) = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	2	[− 10, 10]	0
F24	Powell	U.N	$f(x) = \sum_{i=1}^{D/4}(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$	24	[− 4, 5]	0
F25	Power sum	M.N	$f(x) = \sum_{k=1}^{D}[\left(\sum_{i=1}^{D} x_i^k\right) - b_k]^2$	4	[0, D]	0
F26	Shekel 4.5	M.N	$f(x) = -\sum_{i=1}^{5}[(x - a_i)(x - a_i)^T + c_i]^{-1}$	4	[0, 10]	− 10.1532
F27	Sum square	U.S	$f(x) = \sum_{i=1}^{D} ix_i^2$	30	[− 10, 10]	0
F28	Trid	M.N	$f(x) = \sum_{i=1}^{D}(x_i - 1)^2 - \sum_{i=2}^{D} x_ix_{i-1}$	30	$[-D^2, -D^2]$	− 1500
F29	Zettl	U.N	$f(x) = (x_1^2 + x_2^2 - 2x_1)^2 + 0.25x_1$	2	[− 1, 5]	0.00379
F30	leon	U.N	$f(x) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$	2	[− 1.2, 1.2]	0

M multimodal, *U* unimodal, *S* separable, *N* non separable, *Dim* dimension

Appendix C

See Tables 38, 39, 40, 41 and 42.

Table 38 Unimodal separable function

Function	Name	Formulae	Dim	Range	$f_{\min}$
F1	Step	Description in Appendix B			
F2	Sphere	Description in Appendix B			
F3	Sum squares	Description in Appendix B			
F4	Quartic	Description in Appendix B			

Table 39 Unimodal nonseparable function

Function	Name	Formulae	Dim	Range	$f_{\min}$
F5	Beale	Description in Appendix B			
F6	Easom	Description in Appendix B			
F7	Matyas	Description in Appendix B			
F8	Colville	$f(x) = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90\,(x_3^2 - x_4)^2 + 10.1\,(x_2 - 1)^2 + (x_2 - 1)(x_4 - 1)$	4	[− 10, 10]	0
F9	Zakharov	$f(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^4$	10	[− 5, 10]	0
F10	Schwefel 2.22	Description in Appendix B			
F11	Schwefel 1.2	Description in Appendix B			
F12	Dixon-Price	$f(x) = (x_1 - 1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_i - 1)^2$	30	[− 10, 10]	0

Table 40 Multimodal separable function

Function	Name	Formulae	Dim	Range	$f_{\min}$
F13	Bohachevsky1	Description in Appendix B			
F14	Booth	Description in Appendix B			
F15	Michalewicz2	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(i x_i^2/\pi))^{2m}, m = 10$	2	$[0, \pi]$	−1.8013
F16	Michalewicz5	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(i x_i^2/\pi))^{2m}, m = 10$	5	$[0, \pi]$	−4.6877
F17	Michalewicz10	$f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin(i x_i^2/\pi))^{2m}, m = 10$	10	$[0, \pi]$	−9.6602
F18	Rastrigin	Description in Appendix B			

Table 41 Multimodal nonseparable function

Function	Name	Formulae	Dim	Range	$f_{\min}$
F19	Schaffer	Description in Appendix B			
F20	Six Hump Camel Back	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[− 5, 5]	− 186.73
F21	Boachevsky2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) \times \cos(4\pi x_2) + 0.3$	2	[− 100, 100]	0
F22	Boachevsky3	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos((3\pi x_1) + (4\pi x_2)) + 0.3$	2	[− 100, 100]	0
F23	Shubert	Description in Appendix B			
F24	Rosenbrock	Description in Appendix B			
F25	Griewank	Description in Appendix B			
F26	Ackley	Description in Appendix B			

Table 42 Brief description of CEC 2014 benchmark functions

Function	Name	Dim	Range	$f_{\min}$
F27	Rotated high conditional elliptic function (CEC1)	30	[− 100, 100]	100
F28	Rotated bent cigar function (CEC2)	30	[− 100, 100]	200
F29	Shifted and rotated Rosenbrock's function (CEC4)	30	[− 100, 100]	400
F30	Hybrid function 1 (CEC17)	30	[− 100, 100]	1700
F31	Composition function 1 (CEC23)	30	[− 100, 100]	2300
F32	Composition function 2 (CEC24)	30	[− 100, 100]	2400
F33	Composition function 3 (CEC25)	30	[− 100, 100]	2500

Appendix D

Welded beam design problem

Min $f(x_1, x_2, x_3, x_4) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$

Subject to:

$$g_1(X) = x_1 - x_4 \le 0$$

$$g_2(X) = \delta - 0.25 \le 0$$

$$g_3(X) = \tau - 13600 \le 0$$

$$g_4(X) = \rho - 30000 \le 0$$

$$g_5(X) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5 \le 0$$

$$g_6(X) = 0.125 - x_1 \le 0$$

$$g_7(X) = 6000 - F \le 0$$

where the variables satisfy $0.1 \le x_1, x_4 \le 2.0$ and $0.1 \le x_2, x_3 \le 10$.

$$\rho = 50400/x_3^2x_4$$

$$Q = 6000(14 + x_2/2)$$
$$D = \frac{1}{2}\sqrt{x_2^2 + (x_1 + x_3)^2}\beta = \frac{QD}{J}$$
$$J = \sqrt{2}x_1x_2\,(x_2^2/6 + (x_1 + x_3)^2/2)$$

$$\delta = 65856/3000x_3^3x_4\alpha = 6000/(\sqrt{2}x_1x_2)$$

$$\tau = \sqrt{\alpha^2 + \frac{\alpha\beta x_2}{D} + \beta^2}$$

$$F = 0.61423 \times 10^6 \frac{x_4^3x_3}{6}\left(1 - \frac{x_3\sqrt{30/48}}{28}\right)$$

Pressure vessel design problem

Min $f(x_1, x_2, x_3, x_4) = 0.6224x_1x_2x_3 + 1.7881x_2x_3^2 + 3.1661\,x_4x_1^2 + 19.84\,x_3x_1^2$

Subject to:

$$g_1(X) = -x_1 + 0.0193x_3 \le 0$$

$$g_2(X) = -x_2 + 0.0954x_3 \le 0$$

$$g_3(X) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^2 + 1296000 \le 0$$

$$g_4(X) = x_4 - 240 \le 0$$

where the variables satisfy $0 \le x_1, x_2 \le 100$ and $10 \le x_3, x_4 \le 200$.

Tension/compression spring design problem

Min $f(x_1, x_2, x_3) = (x_3 + 2)x_1^2x_2$

Subject to:

$$g_1(X) = 1 - \frac{x_2^2x_3}{71785x_1^4} \le 0$$

$$g_2(X) = \frac{x_2(4x_2 - x_1)}{12566x_1^3(x_2 - x_1)} + \frac{1}{5108x_1^2} - 1 \le 0$$

$$g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$

$$g_4(X) = \frac{2(x_2 + x_1)}{3} - 1 \le 0$$

where the variables satisfy $0.05 \le x_1 \le 2$, $0.25 \le x_2 \le 1.3$ and

$2 \le x_3 \le 15$.

Gear train design problem

Min $f(x_1, x_2, x_3, x_4) = \left(\frac{1}{6.931} - \frac{x_3x_2}{x_1x_4}\right)^2$

Variable range $12 \le x_1, x_2, x_3, x_4 \le 60$

Three-bar truss design problem

$$\text{Min } f(x_1, x_2) = \left(2\sqrt{2}x_1 + x_2\right) \times l$$

Subject to:

$$g_1(X) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0$$

$$g_2(X) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0$$

$$g_3(X) = \frac{1}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leq 0$$

Variable range $0 \leq x_1, x_2 \leq 1$

Where l =100 cm. $P = 2$ KN/cm^2 $\sigma = 2$KN/cm^2

Cantilever beam design problem

$$\text{Min } f(x_1, x_2, x_3, x_4, x_5) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5)$$

Subject to:

$$g_1(X) = \frac{61}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$$

Variable range $0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100$.

References

1. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. Evol Comput IEEE Trans 1:67–82
2. Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3:95–99
3. Rechenberg I (1973) Evolution strategy: optimization of technical systems through biological evolution, vol 104. Fromman Holzboog, Stuttgart, pp 15–16
4. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. Evol Comput IEEE Trans 3:82–102
5. Ferreira C (2001) Gene expression programming: a new adaptive algorithm for solving problems. Complex Syst 13(2):87–129
6. Ferreira C (2006) Gene expression programming: mathematical modeling by an artificial intelligence. Springer, Berlin **(ISBN 3-540-32796-7)**
7. Koza JR, Rice JP (1992) Genetic programming: the movie. MIT Press, Cambridge
8. Hansen N, Kern S (2004) Evaluating the CMA evolution strategy on multimodal test functions. Springer, Berlin, pp 282–291
9. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
10. Simon D (2008) Biogeography-based optimization. Evol Comput IEEE Trans 12:702–713
11. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the 1995 IEEE international conference on neural networks, pp 1942–1948
12. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell 1:28–39
13. Basturk B, Karaboga D (2006) An artificial bee colony (ABC) algorithm for numeric function optimization. In: Proceedings of the IEEE swarm intelligence symposium, pp 12–14
14. Kevin MP (2002) Biomimicry of bacterial foraging for distributed optimization and control. Control Syst IEEE 22(3):52–67
15. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: Proceedings of the workshop on nature inspired cooperative strategies for optimization (NICSO 2010). Springer, pp 65–74
16. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimization. Int J Bio-Inspired Comput 2:78–84
17. Gandomi AH, Alavi AH (2012) Krill Herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 17(12):4831–4845
18. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings of the world congress on nature & biologically inspired computing, NaBIC 2009, pp 210–214
19. Mucherino A, Seref O (2007) Monkey search: a novel metaheuristic search for global optimization. In: AIP conference proceedings, p 162
20. Du˘san T, Dell'Orco M (2005) Bee colony optimization—a cooperative learning approach to complex transportation problems. In Advanced OR and AI methods in transportation: proceedings of 16th Mini–EURO conference and 10th meeting of EWGT, pp 51–60
21. Chu SA, Tsai PW, Pan JS (2006) Cat swarm optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4099 LNAI 2006, pp 854–858
22. Tang R, Fong S, Yang XS, Deb S (2012) Wolf search algorithm with ephemeral memory. Dig Inf Manag (ICDIM) 2012:165–172
23. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010
24. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
25. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
26. Alireza A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12
27. Mirjalili S, Gandomi AH, Mirjalili SZ, Faris SH,Mirjalili SM (2017)Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Soft 1–29
28. Saremi S, Mirjalili S, Lewis A (2017) Grasshopper optimization algorithm: theory and application. Adv Eng Softw 105:30–47
29. Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23:715–734. https://doi.org/10.1007/s00500-018-3102-4
30. Jain M, Singh V, Rani A (2019) A novel nature-inspired algorithm for optimization: squirrel search algorithm. Swarm Evolut Comput 44:148–175
31. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872. https://doi.org/10.1016/j.future.2019.02.028
32. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43:303–315
33. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76:60–68
34. Fogel D (2009) Artificial intelligence through simulated evolution. Wiley-IEEE Press
35. Glover F (1989) Tabu search—part I. ORSA J Comput 1:190–206
36. Glover F (1990) Tabu search—part II. ORSA J Comput 2:4–32
37. He S, Wu Q, Saunders J (2006) A novel group search optimizer inspired by animal behavioural ecology. In: Proceedings of the 2006 IEEE congress on evolutionary computation, CEC 2006, pp 1272–1278

38. He S, Wu QH, Saunders J (2009) Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans EvolComput13:973–90
39. Atashpaz GE, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Proceedings of the 2007 IEEE congress on evolutionary computation, CEC 2007, pp 4661–4667
40. Kashan AH (2009) League championship algorithm: a new algorithm for numerical function optimization. In: Proceedings of the international conference on soft computing and pattern recognition, SOCPAR'09, pp 43–48
41. Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. Advances in swarm intelligence. Springer, Berlin, pp 355–364
42. Kaveh A (2014) Colliding bodies optimization. Advances in metaheuristic algorithms for optimal design of structures. Springer, Berlin, pp 195–232
43. Gandomi AH (2014) Interior search algorithm (ISA): a novel approach for global optimization. ISA Trans 53(4):1168–1183. https://doi.org/10.1016/j.isatra.2014.03.018
44. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2013) Mine blast algorithm: a new population-based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13:2592–2612
45. Moosavian N, Roodsari BK (2014) Soccer league competition algorithm: a new method for solving systems of nonlinear equations. Int J Intell Sci 4(1):7–16. https://doi.org/10.4236/ijis.2014.41002
46. Dai C, Zhu Y, Chen W (2007) Seeker optimization algorithm. Computational intelligence and security. Springer, Berlin, pp 167–176
47. Ramezani F, Lotfi S (2013) Social-based algorithm (SBA). Appl Soft Comput 13:2837–2856
48. Ghorbani N, Babaei E (2014) Exchange market algorithm. Appl Soft Comput 19:177–187
49. Eita MA, Fahmy MM (2014) Group counseling optimization. Appl Soft Comput 22:585–604
50. Eita MA, Fahmy MM (2010) Group counseling optimization: a novel approach. In: Research and development in intelligent systems XXVI. Springer, London, pp 195–208
51. Xu Y, Cui Z, Zeng J (2010) Social emotional optimization algorithm for nonlinear constrained optimization problems. In: Swarm, evolutionary, and memetic computing. Springer, pp 583–590
52. Huan TT, Kulkarni AJ, Kanesan J (2017) Ideology algorithm: a socio-inspired optimizationmethodology. Neural Comput Appl 28:845–876. https://doi.org/10.1007/s00521-016-2379-4
53. Liu ZZ, Chu DH, Song C, Xue X, Lu BY (2016) Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition. Inf Sci 326:315–333
54. Satapathy SC, Naik A (2016) Social group optimization (SGO): a new population evolutionary optimization technique. Complex Intel Syst 2(3):173–203
55. Naik A, Satapathy SC, Ashour AS, Dey N (2018) Social group optimization for global optimization of multimodal functions and data clustering problems. Neural Comput Appl 30(1):271–287
56. Emami H, Derakhshan F (2015) Election algorithm: a new socio-politically inspired strategy. AI Commun 28(3):591–603
57. Kuo HC, Lin CH (2013) Cultural evolution algorithm for global optimizations and its applications. J Appl Res Technol 11(4):510–522
58. Kulkarni AJ, Durugkar IP, Kumar M (2013) Cohort intelligence: a self supervised learning behavior. In: Systems, man, and cybernetics, SMC, IEEE international conference, IEEE, Manchester, pp 1396–1400
59. Javid AA (2011) Anarchic society optimization: a human-inspired method. In: Evolutionary computation, CE 2011 IEEE congress, IEEE, New Orleans, pp 2586–2592
60. Moghdani R, Salimifard K (2018) Volleyball premier league algorithm. Appl Soft Comput 64:161–185
61. Kumar M, Kulkarni AJ, Satapathy SC (2018) Socio evolution & learning optimization algorithm: a socio-inspired optimization methodology. Future Gener Comput Syst 81:252–272. https://doi.org/10.1016/j.future.2017.10.052
62. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680
63. Webster B, Bernhard PJ (2003) A local search optimization algorithm based on natural principles of gravitation. In: Proceedings of the 2003 international conference on information and knowledge engineering (IKE'03), pp 255–261
64. Erol OK, Eksin I (2006) A new optimization method: big bang–big crunch. Adv Eng Softw 37:106–111
65. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
66. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213:267–289
67. Formato RA (2007) Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog Electromagn Res 77:425–491
68. Alatas B (2011) ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst Appl 38:13170–13180
69. Hatamlou A (2013) Black hole: a new heuristic optimization approach for data clustering. Inf Sci 222:175–184
70. Kaveh A, Khayatazad M (2012) A new meta-heuristic method: ray optimization. Comput Struct 112:283–294
71. Du H, Wu X, Zhuang J (2006) Small-world optimization algorithm for function optimization. Advances in natural computation. Springer, Berlin, pp 264–273
72. Shah-Hosseini H (2011) Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimization. Int J Comput Sci Eng 6:132–140. https://doi.org/10.1504/IJCSE.2011.041221
73. Moghaddam FF, Moghaddam RF, Cheriet M (2012) Curved space optimization: a random search based on general relativity theory. arXiv: 1208.2214
74. Eskandar H, Sadollah A, Bahreininejad A, Hamdi M (2012) Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct. https://doi.org/10.1016/j.compstruc.2012.07.010
75. Tamura K, Yasuda K (2011) Spiral dynamics inspired optimization. J Adv Comput Intell Intell Inform 15(8):1116–1122
76. Rabanal P, Rodríguez I, Rubio F (2007) Using river formation dynamics to design heuristic algorithms. In: Unconventional computation. UC 2007. Lecture notes in computer science, vol 4618. Springer, Berlin, pp 163–177. https://doi.org/https://doi.org/10.1007/978-3-540-73554-0_16
77. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 96:120–133
78. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27:495–513. https://doi.org/10.1007/s00521-015-1870-7
79. Nematollahi AF, Rahiminejad A, Vahidi B (2017) A novel physical based meta-heuristic optimization method known as lightning attachment procedure optimization. Appl Soft Comput 59:596–621. https://doi.org/10.1016/j.asoc.2017.06.033
80. Nematollahi AF, Rahiminejad A, Vahidi B (2020) A novel meta-heuristic optimization method based on golden ratio in nature. Soft Comput 24:1117–1151. https://doi.org/10.1007/s00500-019-03949-w

81. Omidvar MN, Li X, Tang K (2015) Designing benchmark problems for large-scale continuous optimization. Inf Sci 316:419–436
82. Tang K, Li X, Suganthan PN, Yang Z, Weise T (2010) Benchmark functions for the CEC 2010 special session and competition on large-scale global optimization. Technical report, University of Science and Technology of China
83. Wang H, Liang M, Sun C, Zhang G, Xie L (2020) Multiple-strategy learning particle swarm optimization for large-scale optimization problems. Complex Intell Syst. https://doi.org/10.1007/s40747-020-00148-1
84. Gu Q, Li X, Jiang S (2019) Hybrid genetic grey wolf algorithm for large-scale global optimization. Complexity. https://doi.org/10.1155/2019/2653512
85. Hadi AA, Mohamed AW, Jambi KM (2019) LSHADE-SPA memetic framework for solving large-scale optimization problems. Complex Intell Syst 5:25–40. https://doi.org/10.1007/s40747-018-0086-8
86. Qiao W, Yang Z (2019) Solving large-scale function optimization problem by using a new metaheuristic algorithm based on quantum Dolphin swarm algorithm. IEEE Access 7:138972–138989
87. Wen L, Jianjun J, Ximing L, Mingzhu T (2018) Inspired grey wolf optimizer for solving large-scale function optimization problems. Appl Math Model 60:112–126. https://doi.org/10.1016/j.apm.2018.03.005
88. Ran C, Jin Y (2014) A competitive swarm optimizer for large scale optimization. IEEE Trans Cybern 45(2):191–204
89. Ran C, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291:43–60
90. Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach. Part I: solving problems with box constraints. IEEE Trans Evol Comput 18:577–601
91. Lin Q, Liu S, Zhu Q (2018) Particle swarm optimization with a balanceable fitness estimation for manyobjective optimization problems. IEEE Trans Evol Comput 22(4):32–46
92. LiuY QH, Zhang Z, Yao L, Wang C, Mo L, Ouyang S, Li J (2019) A region search evolutionary algorithm for many-objective optimization. Inf Sci 488:19–40. https://doi.org/10.1016/j.ins.2019.03.016
93. Dhiman G, Kumar V (2019) KnRVEA: a hybrid evolutionary algorithm based on knee points and reference vector adaptation strategies for many-objective optimization. Appl Intell 49:2434–2460. https://doi.org/10.1007/s10489-018-1365-1
94. Reddy SR, Dulikravich GS (2019) Many-objective differential evolution optimization based on reference points: NSDE-R. Struct Multidisc Optim 60:1455–1473. https://doi.org/10.1007/s00158-019-02272-0
95. Qin S, Sun C, Zhang G, He X, Tan Y (2020) A modified particle swarm optimization based on decomposition with different ideal points for many-objective optimization problems. Complex Intell Syst. https://doi.org/10.1007/s40747-020-00134-7
96. Matteo MD, Maier HR, Dandy GC (2019) Many-objective portfolio optimization approach for stormwater management project selection encouraging decision-maker buy-in. Environ Model Softw 111:340–355
97. Fang Y, Liu Q, Li M, Laili Y, Pham DT (2019) Evolutionary many-objective optimization for mixed-model disassembly line balancing with multi-robotic workstations. Eur J Oper Res 276(1):160–174. https://doi.org/10.1016/j.ejor.2018.12.035
98. Sun C, Jin Y, Zeng J, Yu Y (2015) A two-layer surrogate-assisted particle swarm optimization algorithm. Soft Comput 19(6):1461–1475. https://doi.org/10.1007/s00500-014-1283-z
99. Sun C, Jin Y, Cheng R, Ding J, Zeng J (2017) Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems. IEEE Trans Evol Comput 21(4):644–660
100. Wang H, Jin Y, Doherty J (2017) Committee-based active learning for surrogate-assisted particle swarm optimization of expensive problems. IEEE Trans Cybern 9:2664–2677
101. Haibo Y, Ying T, Jianchao Z, Chaoli S, Yaochu J (2018) Surrogate-assisted hierarchical particle swarm optimization. Inf Sci. https://doi.org/10.1016/j.ins.2018.04.062
102. Sun C, Ding J, Zeng J (2018) A fitness approximation assisted competitive swarm optimizer for large scale expensive optimization problems. Memet Comput 10:123–134. https://doi.org/10.1007/s12293-016-0199-9
103. Wan K, He C, Camacho A, Shang K, Cheng R, Ishibuchi H (2018) A hybrid surrogate-assisted evolutionary algorithm for computationally expensive many-objective optimization. In: IEEE congress on evolutionary computation (CEC), Wellington, pp 2018–2025
104. Dey N, Rajinikanth V, Ashour A, Tavares JM (2018) Social group Optimization supported segmentation and evaluation of skin melanoma images. Symmetry 10(2):51
105. Rajinikanth V, Satapathy SC (2018) Segmentation of ischemic stroke lesion in brain MRI based on social group optimization and Fuzzy–Tsallis entropy. Arab J Sci Eng 43(8):4365–4378
106. Madhavi G, Harika V (2018) Implementation of social group optimization to economic load dispatch problem. Int J Appl Eng Res 13:11195–11200
107. Monisha R, Mrinalini R, Britto MN (2019) Social group optimization and Shannon's function-based RGB image multi-level thresholding. Smart Intell Comput Appl 105:123–132
108. Praveen SP, Rao KT, Janakiramaiah B (2018) Effective allocation of resources and task scheduling in cloud environment using social group optimization. Arab J Sci Technol 43(8):4265–4272
109. Rao KT (2018) Client-awareness resource allotment and job scheduling in heterogeneous cloud by using social group optimization. Int J Nat Comput Res. https://doi.org/10.4018/IJNCR.2018010102
110. Mafarja M, Aljarah I, Heidari AA, Abdelaziz I, Hammouri FH (2018) Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems. Knowl Based Syst 145:25–45
111. Heidari AA, Faris H, Aljarah I (2019) An efficient hybrid multilayer perceptron neural network with grasshopper optimization. Soft Comput 23:7941–7958. https://doi.org/10.1007/s00500-018-3424-2
112. Jie L, Huiling C, Qian Z (2018) An improved grasshopper optimization algorithm with application to financial stress prediction. Appl Math Model 64:654–668
113. Arora S, Anand P (2019) Chaotic grasshopper optimization algorithm for global optimization. Neural Comput Appl 31:4385–4405. https://doi.org/10.1007/s00521-018-3343-2
114. Aljarah I, Al-Zoubi AM, Faris H (2018) Simultaneous feature selection and support vector machine optimization using the grasshopper optimization algorithm. Cogn Comput 10:478–495. https://doi.org/10.1007/s12559-017-9542-9
115. Mafarja M, Aljarah I, Faris H, Hammouri AI, Al-Zoubi AM, Mirjalili S (2019) Binary grasshopper optimisation algorithm approaches for feature selection problems. Expert Syst Appl 117:267–286
116. Faris H, Mafarja MM, Heidari AA, Ibrahim AI, Mirjalili S, Fujita H (2018) An efficient binary Salp swarm algorithm with crossover scheme for feature selection problems. Knowl Based Syst 154:43–67
117. Abbassi R, Abbassi A, Heidari AA, Mirjalili S (2019) An efficient Salp swarm-inspired algorithm for parameters identification of photovoltaic cell models. Energy Convers Manag 179:362–372
118. Sayed GI, Khoriba G, Haggag MH (2018) A novel chaotic salp swarm algorithm for global optimization and feature selection.

Appl Intell 48:3462–3481. https://doi.org/10.1007/s10489-018-1158-6
119. Attia A, Fergany E (2018) Extracting optimal parameters of PEM fuel cells using Salp swarm optimizer. Renew Energy 119:641–648
120. Ibrahim RA, Ewees AA, Oliva D (2019) Improved Salp swarm algorithm based on particle swarm optimization for feature selection. J Ambient Intell Human Comput 10:3155–3169. https://doi.org/10.1007/s12652-018-1031-9
121. Ibrahim A, Ahmed A, Hussein S, Hassanien AE (2018) Fish image segmentation using Salp swarm algorithm. In: The international conference on advanced machine learning technologies and applications. AMLTA 2018. Advances in intelligent systems and computing. Springer, p 723
122. Nematollahi FA, Rahiminejad A, Vahidi B (2019) A novel multi-objective optimization algorithm based on lightning attachment procedure optimization algorithm. Appl Soft Comput 75:404–427. https://doi.org/10.1016/j.asoc.2018.11.032
123. Shuang S, Zhiwei Y, Lingyu Y, Jun S, Ruoxi W (2018) Wrapper feature selection based on lightning attachment procedure optimization and support vector machine for intrusion detection. In: 2018 IEEE 4th international symposium on wireless systems within the international conferences on intelligent data acquisition and advanced computing systems (IDAACS-SWS). https://doi.org/https://doi.org/10.1109/IDAACS-SWS 8525742
124. Zheng T, Luo W (2019) An enhanced lightning attachment procedure optimization with quasi-opposition-based learning and dimensional search strategies. Comput Intell Neurosci. https://doi.org/10.1155/2019/1589303
125. Mahrous AT, Kamel S, Jurado F, Ebeed M (2020) Optimal power flow solution incorporating a simplified UPFC model using lightning attachment procedure optimization. Electr Energy Syst 30(1):e12170
126. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005 (Kanpur Genetic Algorithms Laboratory, IIT Kanpur)
127. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. Evolut Comput IEEE Trans 3(2):82–102
128. Liang JJ, Qu BY, Suganthan PN, Hernández-Díaz AG (2013)Problem definitions and evaluation criteria for theCEC2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report 12, pp 3–18
129. Liang JJ, Qu BY, Suganthan PN (2013) Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective. Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore Real-Parameter Numerical Optimization
130. Cheng MY, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112
131. Cheng MY, Lien LC (2012) Hybrid artificial intelligence-based PBA for benchmark functions and facility Layout design optimization. J Comput Civ Eng 26(5):612–624
132. Karaboga D, Basturk B (2008) On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput 8(1):687–697
133. Krink T, Filipic B, Fogel GB (2004) Noisy optimization problems—a particular challenge for differential evolution? In: Congress on evolutionary computation, CEC2004, IEEE, vol 1, pp 332–339
134. Surjanovic S, Bingham D (2017) British Columbia, 2015. https://www.sfu.ca/~ssurjano/optimization.html. Accessed 15 Jan 2017
135. Arora JS (2004) Introduction to optimum design. Academic Press.
136. Belegundu AD (1983) Study of mathematical programming methods for structural optimization. Diss Abstr Int Part B Sci Eng 43:1983
137. Coello CA (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41:113–127
138. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2013) Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13:2592–2612
139. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: metaheuristic approach to solve structural optimization problems. Eng Comput 29:17–35