INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Hybridization of Social Group Optimization and Differential Evolution Algorithm for Solving Speed Reducer Design Problem

Sai Shaktimayee Sahu[1]*, **Suresh Chandra Satapathy**[1], **Anima Naik**[2]

1 School of Computer Engineering, KIIT deemed to be University, Bhubaneswar, Odisha, India
2 Raghu Engineering College, Visakhapatnam, Andhra Pradesh, India

OPEN ACCESS

Received: 27-03-2024

Accepted: 22-05-2024

Published: 08-06-2024

Citation: Sahu SS, Satapathy SC, Naik A (2024) Hybridization of Social Group Optimization and Differential Evolution Algorithm for Solving Speed Reducer Design Problem. Indian Journal of Science and Technology 17(23): 2455-2462. https://doi.org/10.17485/IJST/v17i23.965

* **Corresponding author**.

saishakti1997@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: The objective of this study is to present a hybrid approach named SGO-DE, which combines the Social Group Optimization (SGO) algorithm and differential evolution (DE), aiming to balance exploration and exploitation capacities to improve the accuracy of the optimization algorithm in finding optimal solution for speed reducer design problem. This hybrid approach is simulated for a speed reducer mechanical engineering design problem and the results are compared to several other state-of-the-art optimization algorithms. **Method**: To improve the exploration and exploitation of SGO, in its acquiring phase Differential Evolution (DE) is introduced. The individual candidate solutions derived from the Improving phase of SGO tries to acquire better values using DE. This helps in striking a better balance between exploration and exploitation, there by achieving improved optimal values and not getting trapped in local optima. The performance of the SGO-DE method is then evaluated and compared to other optimization algorithms through experimentation on the speed reducer design challenge. **Findings**: The findings of this study indicate that the SGO-DE hybrid approach outperforms other state-of-the-art algorithms by a significant margin in terms of optimization results. The numbers of function evaluations (FEs) significantly go low as less as 6000 compared to other state-of-the algorithms. The comparison demonstrates the efficacy of the SGO-DE method in enhancing solution quality and speeding up execution. **Novelty**: The novelty of this study lies in the development a hybrid approach of SGO and DE which is efficient in achieving competitive performance in less numbers of function evaluation in speed reducer design problem. This hybridization can strike a better balance between exploration and exploitations.

Keywords: SGO; DE; Hybridization; Nature-inspired; Optimization algorithm

1 Introduction

"Nature-inspired population-based optimization algorithms" are meta-heuristic algorithms that largely rely on mimicking artificial intelligence and the natural world. These methods are gradient-free and flexible. Over the past few decades, numerous meta-heuristic optimization strategies have been proposed. The No-Free-Lunch (NFL) theorem states that no algorithm can solve every optimization problem. This theorem inspires far too many researchers to improve upon or combine existing algorithms to tackle a variety of problems, or to propose entirely new algorithms to produce results that can rival those of the existing algorithms. Our hybridization of the SGO [(1)] and DE [(2)] algorithms, which we utilized to solve the speed reducer design (SRD) problem [(3)], an application problem in engineering, was inspired by this idea.

Exploration and exploitation are two essential components of every meta-heuristic optimization technique. The word "exploration" describes the process of searching the entire area and developing new theories with each try. This section shows how a method can carry out a global search. The term "exploitation" refers to the degree of solutions in each iteration. This component illustrates how a method can perform a local search and find the best solution close to an issue. It is important to recognize that these two components are antagonistic. Stated differently, an excessive emphasis on local search, or exploitation, can trap one in local optimal locations, whereas an excessive emphasis on global search, or exploration, can result in a low-quality final best response. An algorithm should be designed in a way that permits balance to determine the optimal solution to the issue. As can be observed from [(4–7)], the SGO algorithm's exploratory search and exploration skills are far superior to many other algorithms', yet it still fails to find the optimal solution for a number of fixed-dimensional multimodal functions. Using the Shekel family (Shekel 5, Shekel 7, Shekel 10) as an illustration. Therefore, it's still important to strike a balance between exploration and exploratory search. We discovered that the SGO algorithm has significant potential for exploitation in [(7)]. We know that the performance of the DE algorithm is significantly influenced by the crossover and mutation operators [(8,9)]. DE mutation operators prefer the search space for both exploration and exploitation. If the mutation rate is too high, the algorithm will be able to explore a larger area of the universe and its exploration capacity will increase. Thus, we can mix the existing SGO and DE algorithms to improve performance as needed. For social group optimization, the SGO-DE is a brand-new hybrid optimization algorithm that combines the benefits of the SGO and DE algorithms. To get around the challenge of creating a speed reducer, we employed the SGO-DE algorithm. The simulation and experimental results presented in this work demonstrate that the SGO-DE algorithm achieves faster convergence and a better optimal solution than the SGO algorithm.

There are several applications for a speed reducer, which is a part of the gear box in a mechanical system. The speed reducer design poses a more challenging benchmark because to its seven design variables and eleven limitations. The objective is to lower the total weight of the speed reducer while adhering to eleven restrictions. Several meta-heuristic methods [(10)] have been proposed in the literature to address this structural optimization problem. However, it has remained a big challenge to get the optimal values of the parameters of the speed reducer problem to arrive at optimal weight. The main challenge happened to be the search space optimizations of design variable. Several metaheuristic algorithms have failed to strike the balance between the local and global search spaces. The balance can be obtained by adjusting the exploration and exploitations characteristics of these algorithms. In this work, SGO is combined with DE in one of the learning phases of SGO to achieve balance between these two characteristics. Exploration drives the algorithms not to exclude the unvisited search space by the algorithm. And exploitation improves the local search capabilities.

The remaining paper is arranged in the following way: In section 2, the proposed SGO-DE hybridization is covered along with a summary of the SGO and DE algorithms. To answer the design problem for a speed reducer, Part 3 displays simulation and testing results. Section 4 serves as the paper's conclusion.

2 Methodology

2.1 SGO algorithm

The SGO algorithm is introduced by S.C. Satapathy & A. Naik [(2)] in the year of 2016. The SGO algorithm is for dealing with the continuous optimization problem. The SGO algorithm is based on social behaviour of human to solve complex problem. Each human is empowered with knowledge that decides the capacity level for solving a problem. The SGO algorithm is a meta-heuristic population-based algorithm, where the population is considered as a group of persons. Each person represents as candidate solution for the problem. The behavioural traits in humans represent the design variables of problems which corresponds to dimension of the problem. Each person acquires knowledge that is corresponding to the 'fitness'. For proper understanding of SGO algorithm, one has to follow the paper [(2)]. The algorithm 1 details the flow of SGO.

Algorithm 1: SGO Algorithm

Start

Initialise population size N, dimension D, self-introspection parameter C, maximum number of iteration max_iter
Create initial population P_i (i = 1, 2,………,N) in D-dimensional search space,
During the initialization phase, randomly distribute all of the group members throughout the search space.
Determine the fitness value f_i for each individual P_i based on the problem at hand.
Step 1: Finding the best member (gbest) of the group
[minvalue, indexno.] =min$\{f_i,\ i = 1, 2, 3, \ldots, N\}$
gbest=P (indexno, :) in the part of a minimization problem.
Step 2: Launch improving phase to update people's knowledge using gbest
Step 3: Launch 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

2.2 DE algorithm

In the year 1997, R. Storn and K. Price[3] introduced the DE algorithm. DE is used to address the challenge of continual optimization. The DE flow is described in algorithm 2.

Algorithm 2: DE Algorithm
Start
Initialize population size N, dimension D, scaling factor F, crossover parameterC_r, maximum number of iteration max_iter
Generate initial population of size N, $X_{i,0} = (x_{i1,0}, x_{i2,0}, x_{i3,0}, \ldots, x_{iD,0})$, i=1, 2,…. N at generation g=0
while (g< max_iter)
For each individual, i, in the population
Generate three random integers, r_1, r_2 and $r_3 \in$ [1, N], with $r_1 \neq r_2 \neq r_3 \neq i$
Generate a random integer $j_{rand} \in$ [1, D]
For each dimension j

$$u_{ij,g} = \{x_{r_1 j,g} + F.\left(x_{r_2 j,g} - x_{r_3 j,g}\right)\ if\ rand_j\,(0,1) \leq C_r or\, j \\ = j_{rand}\, x_{ij,g}, \qquad otherwise$$

End for
Replace $X_{i,g}$, if $U_{i,g}$=$(u_{i1,g}, u_{i2,g}, u_{i3,g}, \ldots, u_{iD,g})$ is better
End for
g=g+1
end while
stop

2.3 Hybridization of SGO and DE (SGO-DE) algorithm

We introduce a low-level relay hybrid (LRH) group in this research, which combines SGO and DE, two meta-heuristics. This novel hybrid algorithm is called the SGO-DE algorithm. The SGO-DE approach is comparable to the SGO algorithm when the appropriate DE algorithm is applied during the acquisition phase. The structure of the SGO-DE algorithm is shown in Figure 1.

3 Results and Discussion

Simulations and experimental results

In this case, the SGO-DE method was applied to solve the SRD problem in mechanical engineering. As a restricted optimization problem, the SRD issue seeks to maximize the speed reducer's weight given the constraints and conditions. The challenge consists of eleven constraints and seven design variables. Every variable is continuous, with the exception of the

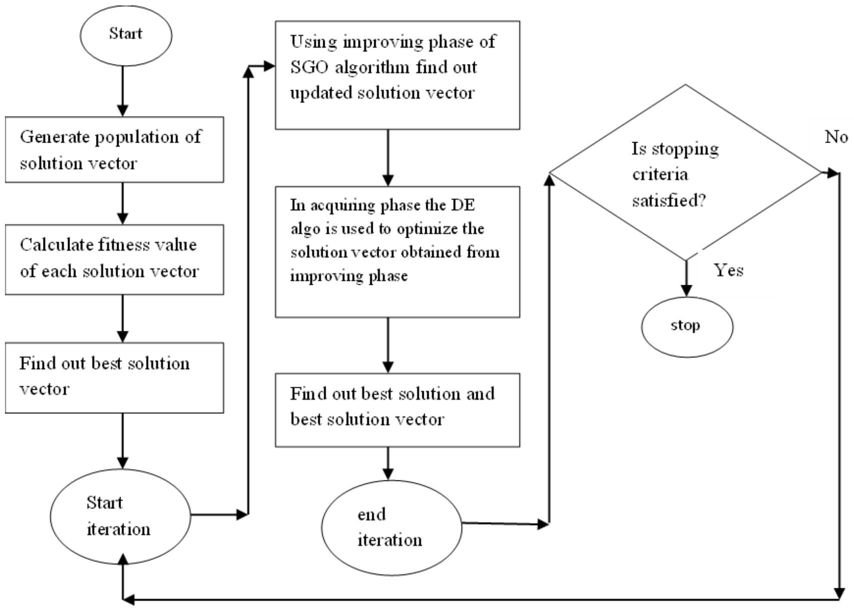

Fig 1. Structure of hybridize approach of SGO and DE (SGO-DE)

third one. Shaft 1 and 2 transverse shaft deflections caused by transmitted force, stresses in shafts 1 and 2, limits on surface stress, and limits on the bending stress of the gear teeth are limited. There have been two occurrences of the problem. A fifth design variable and related boundary condition cause these circumstances. This mixed-integer problem is challenging. The SRD diagram is shown in Figure 2, and the following is the mathematical formulation:

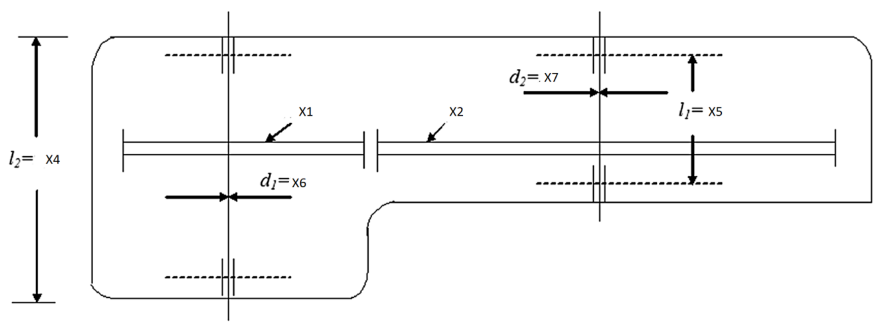

Fig 2. SRD problem

Minimize $f(X) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1\left(x_6^2+x_7^2\right) + 7.4777\left(x_6^3+x_7^3\right) + 0.7854(x_4x_6^2+x_5x_7^2)$

Subject to following constrains

$h_1(X) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$

$h_2(X) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$

$h_3(X) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$

$h_4(X) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$

$h_5(X) = \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 16.9\times10^6} - 1 \leq 0$

$h_6(X) = \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 157.5\times10^6} - 1 \leq 0$

$h_7(X) = \frac{x_2x_3}{40} - 1 \leq 0$

$h_8(X) = \frac{5x_2}{x_1} - 1 \leq 0$
$h_9(X) = \frac{x_1}{12x_2} - 1 \leq 0$
$h_{10}(X) = \frac{1.5x_6+1.9}{x_4} - 1 \leq 0$
$h_{11}(X) = \frac{1.1x_7+1.9}{x_5} - 1 \leq 0$

Part 1. Boundary conditions: 2.6$\leq x_1 \leq 3.6$, 0.7$\leq x_2 \leq 0.8$, 17$\leq x_3 \leq 28$, 7.3$\leq x_4 \leq 8.3$, 7.8$\leq x_5 \leq 8.3$, 2.9$\leq x_6 \leq 3.9$, 5.0$\leq x_7 \leq 5.5$.

Part 2. Boundary conditions are same with part1 except the bound of variables x_5 that change to 7.3 $\leq x_5 \leq 8.3$

Face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of first shaft between bearings x_4, length of second shaft between bearings x_5, diameter of first shaft x_6, and diameter of second shaft x_7, are the design factors in this case.

The SGO and SGO-DE algorithms are built in MATLAB 2016a to compare the performance of each algorithm with that of SGO and other state-of-the-art optimization algorithms. The results of all other methods have been collected from their respective papers, and the experiment was run in a Windows 10 environment with an Intel Core i5, 8 GB of Memory, and these specifications. This is a list of the parameter settings for the SGO and SGO-DE algorithms.

- self-introspection factor C = 0.2. (same for both SGO and SGO-DE)
- crossover parameter C_r=0.1
- scaling factor F=0.9
- population size=30
- maximum iteration=1000

We have divided all our simulations into two parts considering the nature of the algorithms we have shortlisted to compare.

3.1 The simulated results of Part 1:

Table 1 give the simulated outcomes of the SRD problem. Deb's rule was applied in this case by the SGO and SGO-DE algorithms to manage the constraints. The outcomes of the SGO and SGO-DE algorithms are contrasted with those of the following: Simple Evolutionary Strategy (SES), Differential Evolution Strategy (DES), Modified Bacterial Foraging Optimization (MBFOA), League Championship Algorithm (LCA), Improved Accelerated PSO (IAPSO) [1 Upgraded Firefly Algorithm (UFA), ABC, ABC with enhanced food locations (I-ABC greedy)[9–11]. These methods' output and APSO are documented in the corresponding papers. A "NA" in the table indicates that the corresponding document does not include that specific value. Bold facial expressions indicate the best outcome. Table 1 indicates that, in comparison to the previously mentioned method, the SGO algorithm finds the best optimal outcomes. But when compared to all of the meta-heuristics taken into consideration in Table 1, the SGO-DE algorithm finds the most optimal outcome. The SGO and SGO-DE algorithms provide the best solution for the SRD problem, as shown in Table 2. The convergence characteristic of the SGO-DE method used to determine the best solution to the SRD problem in Part 1 is shown in Figure 2. The graphic makes it evident that the SGO-DE algorithm attempts to converge on an optimal solution during the 100 iterations. On the other hand, compared to SGO-DE, SGO requires a significantly higher number of iterations and function evaluations.

Table 1. Results of part 1 of SRD problem

A lgorithms	Worst	Mean	Best	SD	NFEs
SES	3226.2482910	3088.777816	3025.005127	47.36189	36.000
COPSO	NA	2996.4085	2996.3724	2.867e-02	30,000
DES	NA	2996.3480	2996.3480	7.54e-006	36,000
SiC-PSO	NA	2996.34820	2996.3482	0	24,000
MBFOA	NA	3014.759	2999.264	11	30,000
LCA	2996.348165	2996.348165	2996.348165	2.63e-12	24,000
IAPSO	2996.34816497	2996.34816497	2996.34816497	6.88e-13	6000
UFA	2996.3481649986	2996.3481649885	2996.3481649760	4.51e-09	13000
ABC	NA	2997.058412	2997.058412	0	30,000
I-ABC greedy	2996.348165	2996.348165	2996.348165	6.25e-12	6500
APSO	4677.005187	3855.581557	3177.530771	473.767	30,000
SGO	2996.24311502654	2996.243072499899	2996.243040538429	2.60473781e-05	30,000
SGO-DE	**2996.24304047938**	**2996.24304047938**	**2996.24304047938**	**0**	**6000**

Table 2. Optimal solution of part 1 of SRD problem using SGO-DE algorithm

Algorithms	x_1	x_2	x_3	x_4	x_5	x_6	x_7	f(Z)
SGO	3.5000	0.70	17.000	7.300000	7.800000	3.3502146699	5.286517926	2996.243040538429
SGO-DE	3.5000	0.70	17.000	7.300000	7.800000	3.350214666096448	5.286517921717565	2996.24304047938

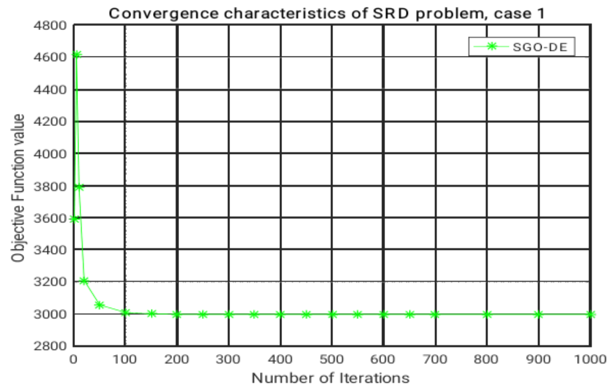

Fig 3. Convergence characteristics of SRD problem for part 1

3.2 The simulated results of Part 2:

The simulation results for the part 2 SRD problem are shown in Table 3. The SGO and SGO-DE algorithms in this case used Deb's rule to handle the constraints. The results of the proposal are compared with (Rank-iMDDE). League Championship Algorithm (LCA) is referenced. (+)-ES, Adaptive constraint-handling approach and Hybrid Evolutionary Algorithm , Improved Accelerated PSO (IAPSO), Parallel Chaotic Local Search Enhanced Harmony Search algorithm (MHS-PCLS), Upgraded Firefly Algorithm (UFA), and Hybrid Evolutionary Algorithm and Adaptive constraint-handling technique (HEAA), Society and Civilization (SC), ABC, DE with Level Comparison (DELC), DE with Dynamic Stochastic Selection (DEDS), Hybridizing PSO with DE (PSO-DE), Mine Blast Algorithm (MBA), Water Cycle Algorithm (WCA) , ABC with Enhanced Food Locations (I-ABC greedy), APSO, and these algorithms' outcomes are documented in the corresponding study. A "NA" in the table indicates that the specified value is absent from the corresponding paper. Bold facial expressions indicate the best outcome. Table 3 shows that, when compared to the previously described methods, the SGO algorithm finds the best optimal outcomes. But when compared to all of the meta-heuristics taken into consideration in Table 3, the SGO-DE algorithm yields the most optimal outcome. The best solution for the SRD problem using several methods is provided in Table 4. The convergence characteristic of the SGO-DE algorithm used in Part 2 to determine the best solution to the SRD problem is shown in Figure 3. The graphic makes it evident that the SGO-DE algorithm attempts to converge on an optimal solution throughout each of the 100 iterations.

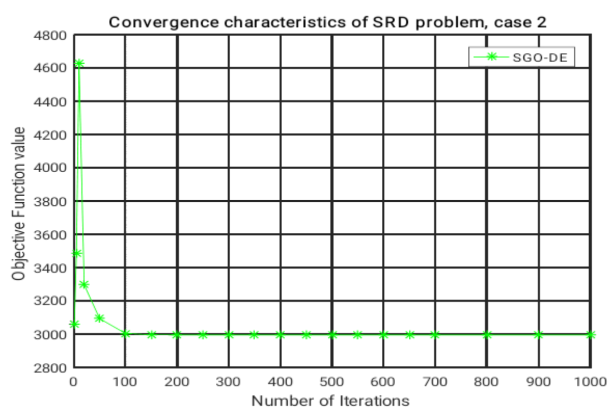

Fig 4. Convergence characteristics of SRD problem for part 2

Table 3. Results of part 2 of SRD problem

Algorithms	Worst	Mean	Best	SD	NFEs
LCA	2994.4710660	2994.4710660	2994.4710660	2.66E-012	24,000
Rank-iMDDE	2994.4710660	2994.4710660	2994.4710660	7.93E-013	19.920
AFA	2996.669016	2996.514874	2996.372698	0.09	50,000
IAPSO	2994.47106615489	2994.47106614777	2.994.47106614598	2.65E-09	6000
MHS-PCLS	2994.471106	2994.4710770	2994.471068	7.142949E-06	10,000
UFA	2994.4710662200	2994.47106618720	2994.4710661647	1.53E-08	6000
$(\mu + \lambda)$-ES	NA	2996.348	2996.348	0	30,000
HEAA	2994.752311	2994.613368	2994.499107	7.0E-02	40,000
SC	3009.964736	3001.758264	2994.744241	4	54,456
ABC	NA	2995.6531	2994.47243	2.98E-12	30,000
DELC	2994.47107	2994.47107	2994.47107	1.9e-12	30,000
DEDS	2994.47107	2994.47107	2994.47107	3.6e-12	30,000
PSO-DE	2996.348204	2996.348174	2996.348167	6.4e-06	54,350
MBA	2999.6524	2996.769	2994.4825	1.56	6300
WCA	2994.505578	2994.474392	2994.471066	7.4E-03	15,150
MDE	NA	2996.36722	2996.356689	8.2E-03	24,000
MBA	2999.6524	2996.769	2994.4825	1.56	6300
I-ABC greedy	2994.902	2994.6631	2994.4710	1.87E-12	6500
APSO	4443.01764	3822.64062	3187.63049	366.146	30,000
SGO	2994.381479512379	2994.3810810554	2994.38103427468	1.2072453e-04	30,000
SGO-DE	**2994.381033749276**	**2994.381033749276**	**2994.381033749276**	**0**	**6,000**

Table 4. Optimal solution of part 2 of SRD problem

Algorithms	x_1	x_2	x_3	x_4	x_5	x_6	x_7	f(Z)
DEDS	3.5	0.7	17	7.3	7.715319	3.350214	5.286654	2994.471
DELC	3.5	0.7	17	7.3	7.715319	3.350214	5.286654	2994.471
HEAA	3.500022	0.7	17	7.300427	7.715377	3.35023	5.286663	2994.499
MDE	3.50001	0.7	17	7.300156	7.800027	3.350221	5.286685	2996.357
PSO-DE	3.5	0.7	17	7.3	7.6	3.350214	5.286683	2996.348
WCA	3.5	0.7	17	7.3	7.711532	3.350214	5.286654	2994.471
MBA	3.5	0.7	17	7.300033	7.715772	3.350218	5.286654	2994.482
LCA	3.5	0.7	17	7.3	7.8	3.350215	5.286683	2994.471
APSO	3.501313	0.7	17	8.127814	8.042121	3.352446	5.287076	3187.63
ABC	3.499999	0.7	17	7.3	7.8	3.350215	5.2878	2997.058
IAPSO	3.5	0.7	17	7.3	7.71532	3.350215	5.286654	2994.471
UFA	3.5	0.7	17	7.3	3.5	3.350215	5.286683	2996.348
MHS-PCLS	3.5	0.7	17	7.3	7.71532	3.350215	5.286655	2994.471
I-ABC greedy	3.50021	0.7	17	7.3	7.715312	3.350215	5.286655	2994.471
SGO	3.5	0. 7	17	7.3	7.71517	3.350215	5.286518	2994.381
SGO-DE	3.5	0.7	17	7.3	7.71517	3.350215	5.286518	2994.381

We conclude from our investigation that the hybrid SGO-DE algorithm outperforms the SGO technique in balancing exploration and exploitation search. After 100 iterations, it independently discovers a more optimal solution and converges faster than SGO.

4 Conclusion and Future Direction

To achieve a balance between the exploration and exploitation of the SGO algorithm, the SGO-DE, a hybrid of the SGO and DE algorithms, is introduced in this study. The low-level relay hybrid (LRH) technology is the foundation of the SGO-DE. The speed reducer design problem is simulated using this hybrid technique. Modern algorithms are contrasted with the outcomes of SGO-DE. The statically simulated solutions to the problems demonstrate the superior performance of the SGO-DE approach over alternative optimizers. In comparison to other methods, it provides as few as 6000 function evaluations. When compared to the algorithms this paper investigated, its efficiency performance is up to five times higher. This accomplishment is mostly attributable to SGO-DE's improved ability to search both local and global search spaces across a wide range of Speed reducer design variable search domains. Furthermore, it is clear from the simulation curve that it has superior convergence to the ideal values. This investigation confirms that SGO-DE is a strategy that shows promise for use in various real-world scenarios. In the future, investigation can be made to use this algorithm in multi-objective optimization problems and find out how the hybridization handles the search space optimization.

References

1) Satapathy S, Naik A. Social group optimization (SGO): a new population evolutionary optimization technique. *Complex & Intelligent Systems*. 2016;2(3):173–203. Available from: https://doi.org/10.1007/s40747-016-0022-8.
2) Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*. 1997;11(4):341–359. Available from: https://doi.org/10.1023/A:1008202821328.
3) Stephen SE, David DCN, GirishDalvi. Design Optimization of Weight of Speed Reducer problem through MatLab and Simulation using ANSYS. *International Journal of Mechanical Engineering and Technology*. 2018;9(8):339–349. Available from: https://iaeme.com/MasterAdmin/Journal_uploads/IJMET/VOLUME_9_ISSUE_8/IJMET_09_08_036.pdf.
4) Naik A, Satapathy SC, Abraham A. Modified social group optimization-a meta-heuristic algorithm to solve short-term hydrothermal scheduling. *Applied Soft Computing*. 2020;95. Available from: https://doi.org/10.1016/j.asoc.2020.106524.
5) Naik A, Satapathy SC. A comparative study of social group optimization with a few recent optimization algorithms. *Complex & Intelligent Systems*. 2021;7(1):249–295. Available from: https://doi.org/10.1007/s40747-020-00189-6.
6) Bilal, Pant M, Zaheer H, Garcia-Hernandez L, Abraham A. Differential Evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*. 2020;90. Available from: https://doi.org/10.1016/j.engappai.2020.103479.
7) Dhanalakshmy DM, Akhila MS, Vidhya CR, Jeyakumar G. Improving the search efficiency of differential evolution algorithm by population diversity analysis and adaptation of mutation step sizes. *International Journal of Advanced Intelligence Paradigms*. 2020;15(2):119–145. Available from: https://doi.org/10.1504/IJAIP.2020.105141.
8) Sahu SS, Satapathy SC. Improvement of Modified Social Group Optimization (MSGO) Algorithm for Solving Optimization Problems. In: Intelligent Computing and Communication. ICICC 2022. Advances in Intelligent Systems and Computing;vol. 1447 of Advances in Intelligent Systems and Computing. Singapore. Springer. 2023;p. 631–636. Available from: https://doi.org/10.1007/978-981-99-1588-0_55.
9) Azqandi MS, Delavar M, Arjmand M. An enhanced time evolutionary optimization for solving engineering design problems. *Engineering with Computers*. 2020;36:763–781. Available from: https://doi.org/10.1007/s00366-019-00729-w.
10) Chen H, Xu Y, Wang M, Zhao X. A balanced whale optimization algorithm for constrained engineering design problems. *Applied Mathematical Modelling*. 2019;71:45–59. Available from: https://doi.org/10.1016/j.apm.2019.02.004.
11) Kalananda VKRA, Komanapalli VLN. A combinatorial social group whale optimization algorithm for numerical and engineering optimization problems. *Applied Soft Computing*. 2021;99. Available from: https://doi.org/10.1016/j.asoc.2020.106903.