Sādhanā (2021)46:121
https://doi.org/10.1007/s12046-021-01641-0

A partition cum unification based genetic- firefly algorithm for single objective optimization

DHRUBAJYOTI GUPTA, ANANDA RABI DHAR and SHIBENDU SHEKHAR ROY*

Department of Mechanical Engineering, National Institute of Technology Durgapur, West Bengal, India
e-mail: dg.17u10024@btech.nitdgp.ac.in; ard.16me1104@phd.nitdgp.ac.in; shibendu.roy@me.nitdgp.ac.in

MS received 14 February 2021; revised 9 April 2021; accepted 19 April 2021

Abstract. Firefly algorithm is one of the most promising population-based meta-heuristic algorithms. It has been successfully applied in many optimization problems. Several modifications have been proposed to the original algorithm to boost the performance in terms of accuracy and speed of convergence. This work proposes a partition cum unification based genetic firefly algorithm to explore the benefits of both the algorithms in a novel way. With this, the initial population is partitioned into two compartments based on a weight factor. An improved firefly algorithm runs in the first compartment, whereas, the genetic operators like selection, crossover, and mutation are applied on the relatively inferior fireflies in the second compartment giving added exploration abilities to the weaker solutions. Finally, unification is applied on the subsets of fireflies of the two compartments before going to the next iterative cycle. The new algorithm in three variants of weightage factor have been compared with the two constituents i.e. standard firefly algorithm and genetic algorithm, additionally with some state-of-the-art meta-heuristics namely particle swarm optimization, cuckoo search, flower pollination algorithm, pathfinder algorithm and bio-geography based optimization on 19 benchmark objective functions covering different dimensionality of the problems viz. 2-D, 16-D, and 32-D. The new algorithm is also tested on two classical engineering optimization problems namely tension-compression spring and three bar truss problem and the results are compared with all the other algorithms. Non-parametric statistical tests, namely Wilcoxon rank-sum tests are conducted to check any significant deviations in the repeated independent trials with each algorithm. Multi criteria decision making tool is applied to statistically determine the best performing algorithm given the different test scenarios. The results show that the new algorithm produces the best objective function value for almost all the functions including the engineering problems and it is way much faster than the standard firefly algorithm.

Keywords. Meta-heuristic algorithms; evolutionary computing; firefly algorithm; genetic algorithm; hybridization; global optimization.

1. Introduction

Meta-heuristic algorithms are approximate techniques for solving an optimization problem. They capitalize on intuitions along with randomness property in their search for improving solutions at hand through successive iterations. Nature has inspired the development of many meta-heuristic algorithms [1, 2]. The earliest of this kind, in the form of genetic algorithm (GA) [3, 4] was motivated by Darwin's survival of the fittest by means of natural selection. Several researchers have shown there keen interests in this beautiful domain since then. As a result, numerous meta-heuristic algorithms have emerged over years [5, 6].

Different creatures communicate with each other through various unique modes of communication. Fireflies use their flashing lights of varying intensity to communicate for attracting mates or warning predators. A suitable mate would either communicate back mimicking the same pattern of flash or would produce response in a different flashing pattern. The intensity of the light which is the key source of attraction also depends on the distance between the source and the observer because of absorption in air. Environmental conditions like wind, darkness or fogginess also regulate the intensity. This phenomenon besides soothing the eyes of nature lovers in the darkness of summer nights has become the epitome of inspiration for many scientific researchers mainly dealing with optimization problems.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s12046-021-01641-0.

*For correspondence
Published online: 17 June 2021

1.1 *Standard firefly algorithm*

Fireflies may be conceived as candidate solutions in the search space. The attraction and movements offer heuristic goals to achieve brighter fireflies i.e. better solutions. The standard firefly algorithm (FA) which has been developed by Yang [7] has two parts in the position updates as shown in Equation (1),

$$x_i = x_i + \beta_0^{(-\gamma \times r_{ij}^2)}(x_j - x_i) + \alpha(\varepsilon() - 0.5) \qquad (1)$$

where, x_i is the position of the firefly that is to be updated based on the difference of position of another firefly denoted by x_j; in the first part, which is driven by attraction heuristic, β_0 is the light intensity when the distance between them is 0, γ is the absorption coefficient and r_{ij} is the Euclidean distance between them; in the second part, which models random movement, α is the coefficient of randomness and $\varepsilon()$ yields a random value in [0, 1].

1.2 *Modifications of standard FA*

FA is prone to premature convergence triggering efforts to relax the constant parameters used therein. Studies also show that this is comparatively slow and often suffers from problems of falling into local minima [8]. The movement update function solely depends on the present light intensity or fitness and no previous memory is preserved. Furthermore, the algorithm parameters are fixed for all the successive iterations making the step length of learning, constant, thereby foregoing the needed exploitation especially at the later stages. Enormous efforts have been channelled to boost the performance of the standard FA [9, 10]. The various modifications, in accordance with some broad categories are as follows.

1.2a *Based on adaptive parameters*: In this category, the parameters i.e. the user defined constants, used in like any other meta-heuristic algorithms, have been updated in the successive iterations. These constants control the degree of exploitation and exploration. The randomness coefficient α has been tuned based on the current iteration number by Shakarami and Sedaghati [11]. Kavousi-Fard *et al* [12] have improvised the randomness part using crossover and mutation operators. Modification of attraction related parameters (β_0, γ and r_{ij}) have been also explored by many researchers. Lin *et al* [13] have used a new concept of virtual distance, r', and computed the intensity of light β as $\beta_0\gamma(1 - r')$ to update the position of fireflies. Gandomi *et al* [14] have employed 12 different chaos functions which have non-repetition and ergodicity properties for updating the constant values of the algorithm (β and γ). Sulaiman *et al* [15] have introduced minimum variance distance replacing the Cartesian distance for the attraction part. The random part has been modified with a mutation operator. Wang *et al* [16] have handled the premature convergence by regulating all the constant parameters based on the difference of light intensity measured by a formula dependent on number of iteration. Yu *et al* [17] have proposed wise step length control specific to individual firefly based on its personal and global best position for updating its current position. A practical problem of optimizing droplet ejection speed in electro-hydrodynamic inkjet printing has been reportedly solved by using an improved FA [18]. The authors have used a new rule for updating the brightness instead of a constant initial brightness coefficient for getting comparatively better results.

Keeping in view the simple modifications in this approach, the results have been really encouraging. It has been proved that keeping updating strategy same may lead to stagnation. None of the cases in this line have considered the need for changing updating strategy in general. This method has also not considered the past memory. This method has not been able to reduce the search space to speed-up the convergence.

1.2b *Based on update strategy*: The second class of approaches for modification deals with changing the strategy of updating the position of the fireflies. Opposition based approach has been introduced by Roy *et al* [19] for solving high dimensional problems where opposite or reverse set of regularly achieved population has been determined in each iteration. The brightest one has been updated by $x_{min} + x_{max} - x$ so that it can still improve from the initial local extremes. This method to some extent mimics the mutation operator in genetic algorithm which has eventually initiated many experiments with different methods and levels of incorporation. Yu *et al* [20] have defined a criterion of hazardous condition based on which a mutation can be operated on the weaker solutions to improve them. Additionally, the authors have preserved the historical performance in memory so that hitherto best cases are not left-out. Kazemzadeh-Persi [21] has added k newborn generations in each iteration with the help of mutation. The authors also have suggested updating the position based on the average directions of movement dictated by all the brighter fireflies. Another line of modification has been approached [12, 22] by employing five crossover operators and three mutation operators along with adaptive α. Three random solutions have been considered for generating the mutated solutions, one by applying $x_{mute1} = x_{q1} + \varepsilon(x_{q2} - x_{q3})$, and another for any iteration number t by applying $x_{mute2} = x_{mute1} + \varepsilon t(x_b - x_w)$. In other similar works [11, 23], the value of α has been made adaptive by chaotic mapping or applying mutation operator. Mohammadi *et al* [23] have proposed the update of the position with respect to the brighter firefly x_j, thereby adding more exploration capability. Hassanzadeh and Kanan [24] have used fuzzy attraction function represented by a Cauchy distribution from the selected top k brighter fireflies to impose the influence of a collection rather than an individual in order to explore towards the global best. In

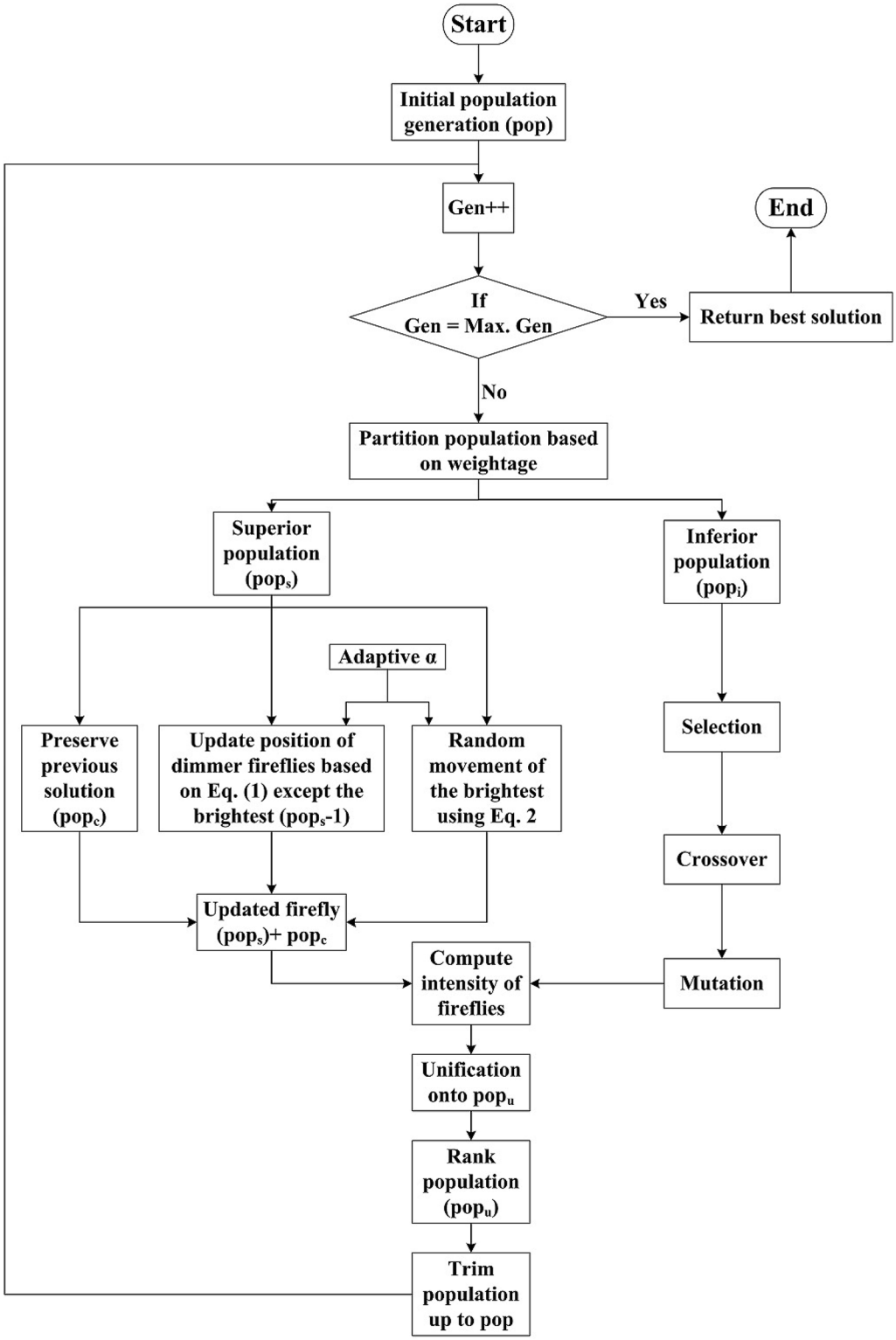

Figure 1. Flowchart of the new algorithm.

Table 1. Benchmark objective functions.

Sl No	Function Name	Equation	Type	Range
1	Ackley (1)	$-20e^{-0.02\sqrt{D^{-1}\sum_{i=1}^{D}x_i^2}} + e^{D^{-1}\sum_{i=1}^{D}\cos(2\pi x_i)} + 20 + e$	continuous, differentiable, non-separable, scalable, multimodal	[-35,35]
2	Beale	$(1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	continuous, differentiable, non-separable,non-scalable, unimodal	[-4.5,4.5]
3	Carrom Table	$-[\cos(x_1)\cos(x_2)\exp\lvert(x_1^2 + x_2^2)^{0.5}/\pi\rvert]^2/30$	continuous, differentiable, non-separable, non-scalable, multimodal	[-10,10]
4	Easom	$-\cos(x_1)\cos(x_2)\exp[-(x_1 - \pi)^2 - (x_2 - \pi)^2]$	continuous, differentiable, separable, non-scalable, multimodal	[-100,100]
5	Goldstein-Price	$[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	continuous, differentiable, non-separable, non-scalable, multimodal	[-2,2]
6	Michalewicz	$-\sum_{i=1}^{D}(\sin(x_i)\sin^{20}(\frac{ix_i^2}{\pi}))$	continuous, differentiable, separable, scalable, multiimodal	$[0,\pi]$
7	Schaffer (6)	$\sum_{i=1}^{D-1}[0.5 + \frac{\sin^2\sqrt{x_i^2+x_{i+1}^2} - 0.5}{[1+(0.001(x_i^2+x_{i+1}^2))]^2}]$	continuous, differentiable, non-separable, scalable, multimodal	[-100,100]
8	Xin-She-Yang (3)	$exp(-\sum_{i=1}^{D}(\frac{x_i}{15})^{10}) - 2exp(-\sum_{i=1}^{D}x_i^2)\prod_{i=1}^{D}cos^2(x_i)$	continuous, differentiable, non-separable, scalable, unimodal	[-20,20]
9	Zettl	$(x_1^2 + x^2 - 2x_1)^2 + 0.25x_1$	continuous, differentiable, non-separable, non-scalable, unimodal	[-5,10]
10	Zakharov	$\sum_{i=1}^{D}x_i{}^2 + \sum_{i=1}^{D}(0.5ix_i)^2 + \sum_{i=1}^{D}(0.5ix_i)^4$	continuous, differentiable, separable, scalable, unimodal	[-5,10]
11	Alpine (1)	$\sum_{i=1}^{D}\lvert x_i\sin(x_i) + 0.1x_i\rvert$	continuous, non-differentiable, separable, non-scalable, multimodal	[-10,10]
12	Dixon-Price	$(x_i - 1)^2 + \sum_{i=2}^{D}i(2x_i^2 - x_{i-1})^2$	continuous, differentiable, non-separable, scalable, unimodal	[-10,10]
13	Levy	$\sin^2(\pi(1 + \frac{x_1-1}{4})) + \sum_{i=1}^{D-1}(\frac{x_i-1}{4})^2[1 + 10\sin^2(1 + \pi(1 + \frac{x_i-1}{4}))] + (\frac{x_d-1}{4})^2[1 + \sin^2(2\pi(1 + \frac{x_d-1}{4}))]$	continuous, differentiable, separable, scalable, multimodal	[-10,10]
14	Rastrigin	$10D + \sum_{i=1}^{D}(x_i^2 - 10cos(2\pi x_i))$	continuous, differentiable, separable, scalable, multiimodal	[-5.12,5.12]
15	Rosenbrock	$\sum_{i=1}^{D}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	continuous, differentiable, non-separable, scalable, multiimodal	[-30,30]
16	Sphere	$\sum_{i=1}^{D}x_i^2$	continuous, differentiable, separable, scalable, unimodal	[0,10]
17	Step (2)	$\sum_{i=1}^{D}(\lfloor x_i + 0.5\rfloor)^2$	discontinuous, non-differentiable, separable, scalable, unimodal	[-100,100]
18	Sum Squares	$\sum_{i=1}^{D}ix_i^2$	continuous, differentiable, separable, scalable, unimodal	[-100,100]
19	Xin-She-Yang (1)	$\sum_{i=1}^{D}\varepsilon_i\lvert x_i\rvert^i$	continuous, non-differentiable, separable, scalable, multimodal	[-5,5]

Table 2. Parameter settings for the algorithms.

Algorithm	Parameter	Value
GA	Crossover probability p_c	0.7
	Mutation probability p_m	0.3
FA	α	4.0
	β	1.0
	γ	2.0
	m	2.0
PSO	Inertia factor w	0.9
	c_1	1
	c_2	1
CS	Discovery rate of alien solutions pa	0.25
FPA	Probability switch p	0.8
PFA	N/A	
BBO	Habitat modification probability	1
	Immigration probability limits	[0 1]
	Step size	1
	Max immigration (I) and Max emigration (E)	1
	Mutation probability p_m	0.001

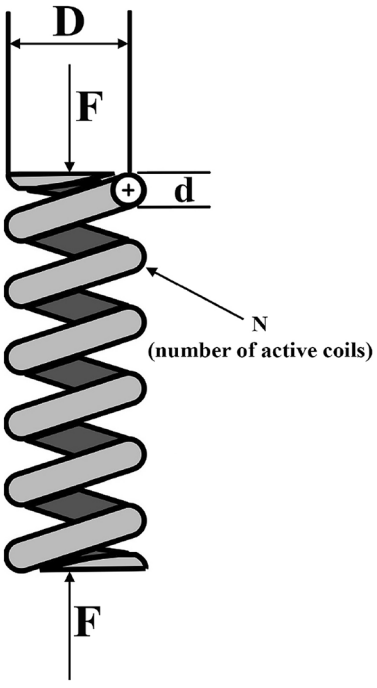

Figure 2. Tension-compression spring mass problem.

some literature [25, 26] , mutation has been used replacing the updating formula in Equation (1) or as an additional operator immediately after using the same updating formula. Wang *et al* [27] have introduced Cauchy distribution

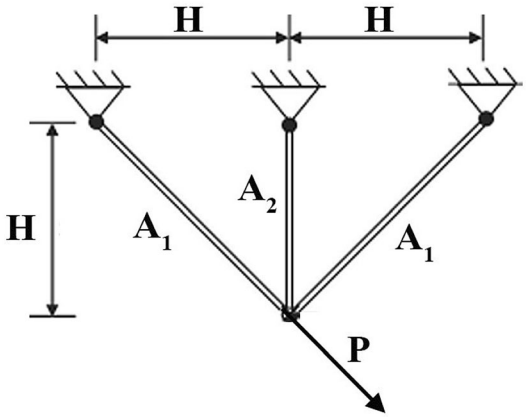

Figure 3. 3-bar truss structure problem.

for performing mutation on the initially fitter fireflies known as good nodes set to obtain improved results on benchmark functions. In another recent work by Peng *et al* [28], a novel courtship learning strategy has been adopted where the updating of the position of the lower intensity male fireflies have been accomplished using the guidance from the relatively superior female fireflies. Tong *et al* [29] have proposed using several diverse sub-groups of FA and exchanging information for learning collectively about the global best more effectively. Zhou *et al* [30] have adopted partial attraction model which have been used to protect swarm diversity and utilize full individual information. Baykasouglu *et al* [31] have proposed a modification to FA to solve dynamic multidimensional knapsack problem by replacing pair-wise comparison by a specialized comparison based on a dynamic variable in each iteration. The authors have also used a probabilistic switch to decide whether to update the position or not for the firefly to avoid early convergence and reduce computational time.

This method of modification has been deemed very useful to specific problems. This has not considered the already improvement areas like adaptive control or changing the randomness characteristics for better explorations. This method has been not able to reduce the search space and the time complexity of convergence.

1.2c *Based on solution space and randomness control*: In the third category of approaches, two types of modification have been tried. Firstly, changing the solution space to an easy search space has been explored by Liu *et al* [32]. They have used quaternion representation for $x_i(k) = (y_1^i, y_2^i, y_3^i, y_4^i)$ for all components k. The update of position takes place in the quadruple-folded search space, while the computation of brightness is done in actual solution space using a norm transform function. Fister *et al* [33] have also used quaternion representation of the solution space for

Table 3. Comparison results of 2-D functions (function 1 - 10).

Function Name	Ackley(1) $f^* = 0$		Beale $f^* = 0$		Carrorm Table $f^* = -24.1568$		Easom $f^* = -1$		Goldstein-Price $f^* = 3$	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	864.69 ± 103.9	1.66961 ± 1.981	857.21 ± 156.93	0.06271 ± 0.275	755.98 ± 23.89	-24.11448 ± 0.661	775.91 ± 28.55	-0.28769 ± 0	880.65 ± 40.8	25.94052 ± 12.067
FPA	491.19 ± 35.01	0.02603 ± 0.034	514.12 ± 9.34	0.03161 ± 0.013	470.24 ± 223.17	-24.14302 ± 0.01	437.33 ± 35.46	-0.9963 ± 0.003	533.07 ± 182.44	3.71077 ± 0.44
PFA	5577.09 ± 99.56	**0**.00796 ± **0**.676	5513.8 ± 327.41	0.49529 ± 0.43	5455.42 ± 325.17	-22.32625 ± 1.217	5494.31 ± 345.29	-0.9753 ± 0.199	5563.64 ± 310.11	4.12467 ± 2.478
CS	144.12 ± 10.91	0.6378 ± 0.066	155.08 ± 13.16	0.01743 ± 0.001	140.13 ± 169.79	-23.98976 ± 0.139	148.6 ± 15.46	0 ± 0.033	161.07 ± 8.72	3.56986 ± 8.62
PSO	282.74 ± 12.34	0.15372 ± 0.444	332.61 ± 221.95	0.00066 ± 0.036	283.24 ± 3.47	-23.73141 ± 0.261	266.29 ± 12.93	-0.85847 ± 0.151	367.02 ± 87.38	**3**.01595 ± **1**.283
GA	448.8 ± 22.29	0.08783 ± 0.042	499.66 ± 292.46	**0**.00023 ± **0**.003	452.29 ± 26.07	-22.51536 ± 0.805	470.24 ± 178.27	-0.90932 ± 0.283	565.99 ± 10.15	3.03793 ± 0.059
FA1	3530.06 ± 33.52	0.18251 ± 2.448	3567.96 ± 295.86	0.00052 ± 0.001	3550.03 ± 266.92	-24.07527 ± 6.409	716.58 ± 127.07	0 ± 0	3537.54 ± 260.74	3.03903 ± 0.017
FA2	712.61 ± 38.42	0.04547 ± 0.009	748.01 ± 8.41	0.00042 ± 0	701.14 ± 188.39	-24.14954 ± 7.337	705.61 ± 180.58	**-0**.99925 ± **0**.001	781.41 ± 149.18	3.02257 ± 0.049
FA3	1263.61 ± 17.63	**0**.01561 ± **0**.013	1250.16 ± 181.63	**0**.00017 ± **0**	1237.69 ± 178.26	**-24**.15449 ± **8**.405	1181.34 ± 213.6	-0.99875 ± 0	1292.55 ± 207.56	3.03392 ± 0.028
FA4	1959.79 ± 28.02	0.02318 ± 0.008	1989.68 ± 269.35	0.00058 ± 0	1925.86 ± 183.05	**-24**.15507 ± **7**.337	1848.56 ± 205.93	**-0**.99978 ± **0**.001	1939.32 ± 206.32	**3**.01214 ± **0**.037

Function Name	Michalewicz $f^* = -1.8013$		Schaffer(6) $f^* = 0$		Xin-She-Yang(3) $f^* = -1$		Zettl $f^* = -0.00379$		Zakharov $f^* = 0$	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	780.9 ± 252.33	-1.77034 ± 0.152	810.34 ± 249	0.07721 ± 0.07	912.56 ± 33.96	0 ± 0	763.46 ± 249.89	0.17228 ± 0.181	934.49 ± 161.28	0.27375 ± 0.319
FPA	469.74 ± 11.27	**-1**.78852 ± **0**.041	469.74 ± 6.3	**0** ± **0**	713.59 ± 29.15	0 ± 0.431	455.77 ± 12.04	0.00374 ± 0.009	613.86 ± 7.91	0.01474 ± 0.01
PFA	5565.12 ± 321.64	-1.71489 ± 0.23	5557.66 ± 429.56	0 ± 0.074	5639.92 ± 311.22	-0.22795 ± 0	5448.44 ± 335.01	-0.00075 ± 0.019	5693.3 ± 290.4	**0** ± **0**.089
CS	144.12 ± 196.47	-1.77801 ± 0.001	173.52 ± 9.1	0.06781 ± 0.009	399.93 ± 9.42	-0.26711 ± 0.305	136.62 ± 9.18	0.04544 ± 0.008	377.49 ± 12.73	0.03014 ± 0.002
PSO	283.24 ± 5.57	-1.78886 ± 0.064	303.19 ± 6.98	0.00352 ± 0.053	522.1 ± 3.35	-0.48026 ± 0.293	268.78 ± 5.45	-0.00342 ± 0.043	406.89 ± 6.84	0.00082 ± 0.099
GA	466.25 ± 22.95	-1.79879 ± 0.001	524.1 ± 9.56	0.00951 ± 0.007	949.48 ± 170.5	-0.96399 ± 0.092	435.34 ± 218.34	-0.00169 ± 0.001	907.08 ± 23.57	0.00384 ± 0.003
FA1	3504.15 ± 263.56	-1.79212 ± 0.008	3533.07 ± 310.87	0.08209 ± 0.095	6364.99 ± 267.68	-0.25622 ± 0.284	3473.23 ± 306.41	-0.0035 ± 0	6753.45 ± 328.74	0.00026 ± 0
FA2	715.09 ± 219.01	-1.7982 ± 0.004	726.06 ± 19.28	0.00352 ± 0.005	1344.39 ± 65.76	-0.9984 ± 0.002	687.66 ± 10.41	**-0**.00376 ± **0**	1334.41 ± 16.67	0.00036 ± 0
FA3	1243.18 ± 209.2	-1.79738 ± 0.003	1283.55 ± 215.15	**0** ± **0**.004	2314.81 ± 175.82	**-0**.99894 ± **0**	1179.85 ± 208.87	**-0**.00365 ± **0**	2322.29 ± 183.66	0.00081 ± 0
FA4	1981.7 ± 193.66	**-1**.79887 ± **0**.003	1994.19 ± 186.01	0.00837 ± 0.01	3674.16 ± 447.11	**-0**.99973 ± **0**.063	1881.49 ± 202.7	-0.0037 ± 0	3694.13 ± 225.41	**0**.00005 ± **0**

enhancing performance and avoiding stagnation. Wang *et. al*. [34] have used a predefined neighbourhood to overcome the problem of oscillation based on too much attractions happening during the search. Secondly, the randomness behaviour of the fireflies has been controlled by following certain probability distribution functions. For example, Farahani *et al* [35] have introduced Gaussian distribution, whereas Yang [36] has used Levy distribution for generating component wise product with the original random fraction. The randomness property has been improved by adding direction of movement of the fireflies using an additional sign vector in case of the later. Tighzert *et al* [37] have presented a set of new compact firefly algorithms involving levy functions, elitism or preservation of previous performances and opposition-based learning. They have tested on benchmark functions considering 30 dimensions. Wu *et al* [38] have also used adaptive logarithmic Levy distribution to improve over the global searching characteristics of FA using a probabilistic switch.

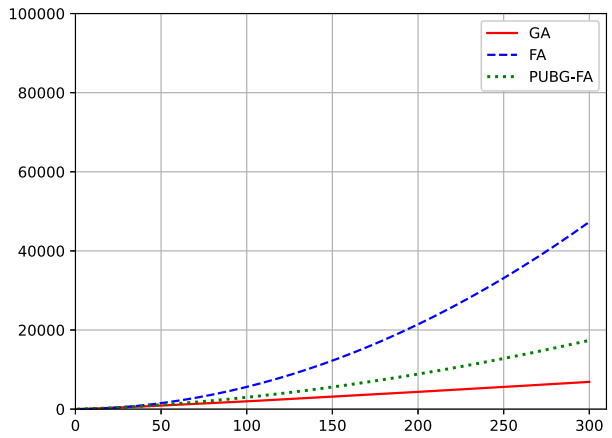

Figure 4. Time complexity plot of the new algorithms and constituents.

The best part about this method is the reduction of search space thereby minimizing the computational cost of the algorithm. This approach of modification has been proved beneficial in most of the cases though the use of other lines of modifications like adaptive control and changing the updating strategy along with it would have clearly achieved more robustness.

1.2d *Hybrid approaches*: There have been a few attempts of hybridization of the FA with other evolutionary algorithms. Luthra and Pal [39] have used genetic operators like crossover along with the standard FA for solving a monoalphabatic substitution cipher problem. Rahmani and MirHassani [40] have applied genetic crossover operator on the two fittest fireflies and mutation on the whole population based on the mutation probability after the pass of standard algorithm in each iteration. They have used this

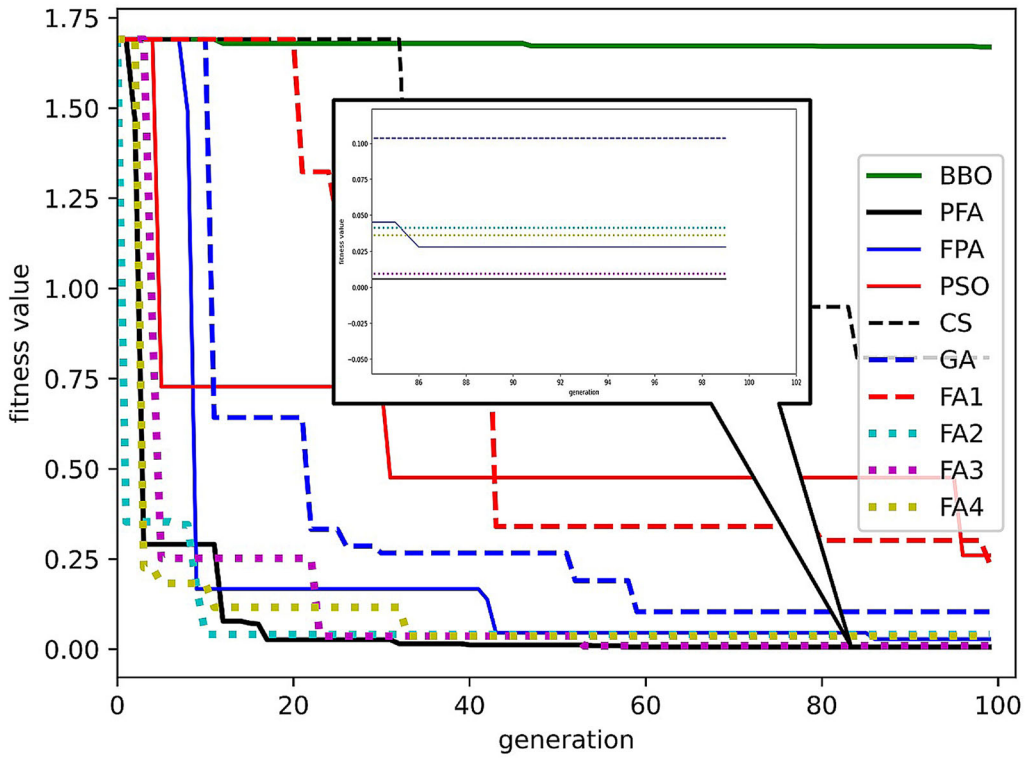

Figure 5. Performance plot of Ackley (1) Function.

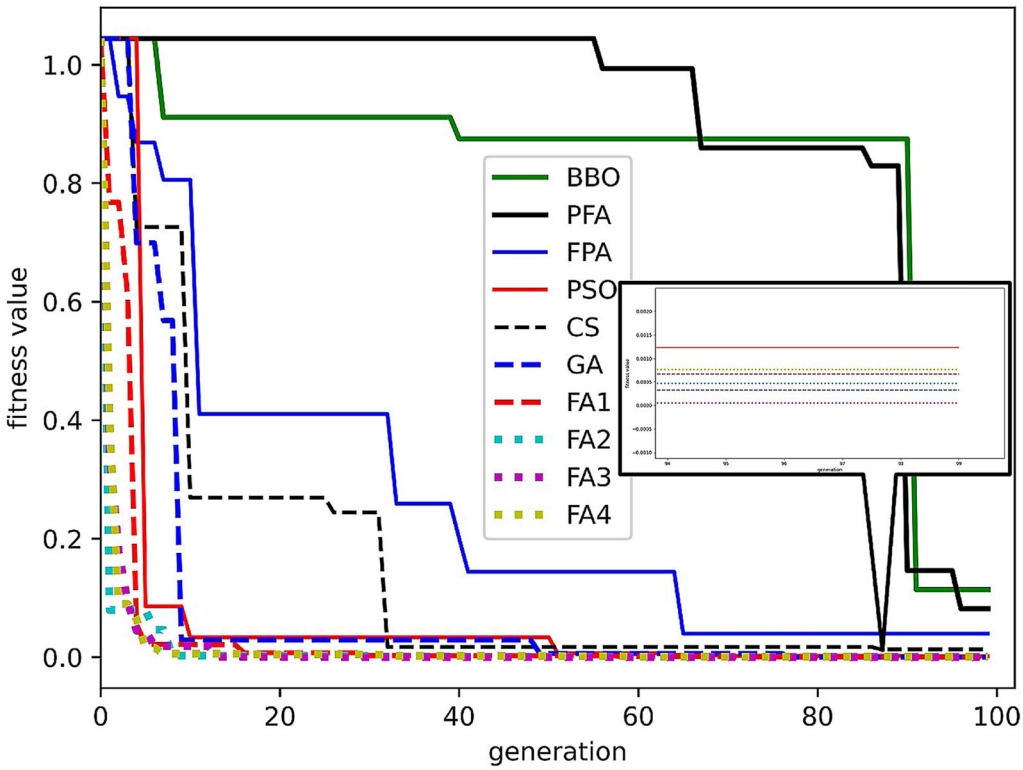

Figure 6. Performance plot of Beale Function.

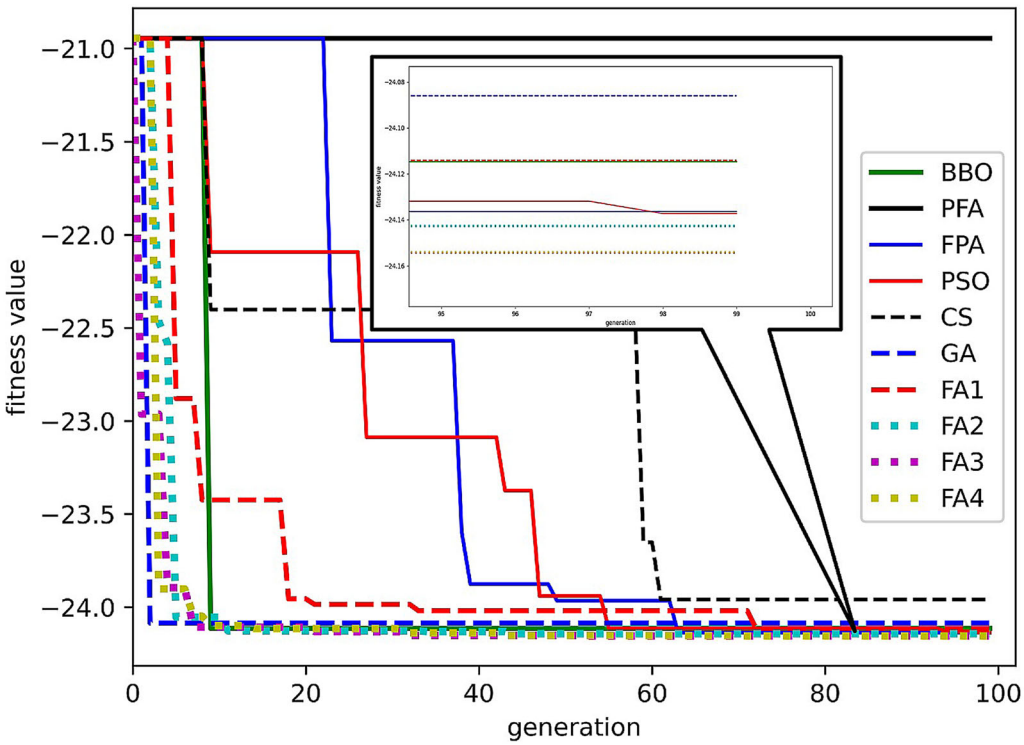

Figure 7. Performance plot of Carrom Table Function.

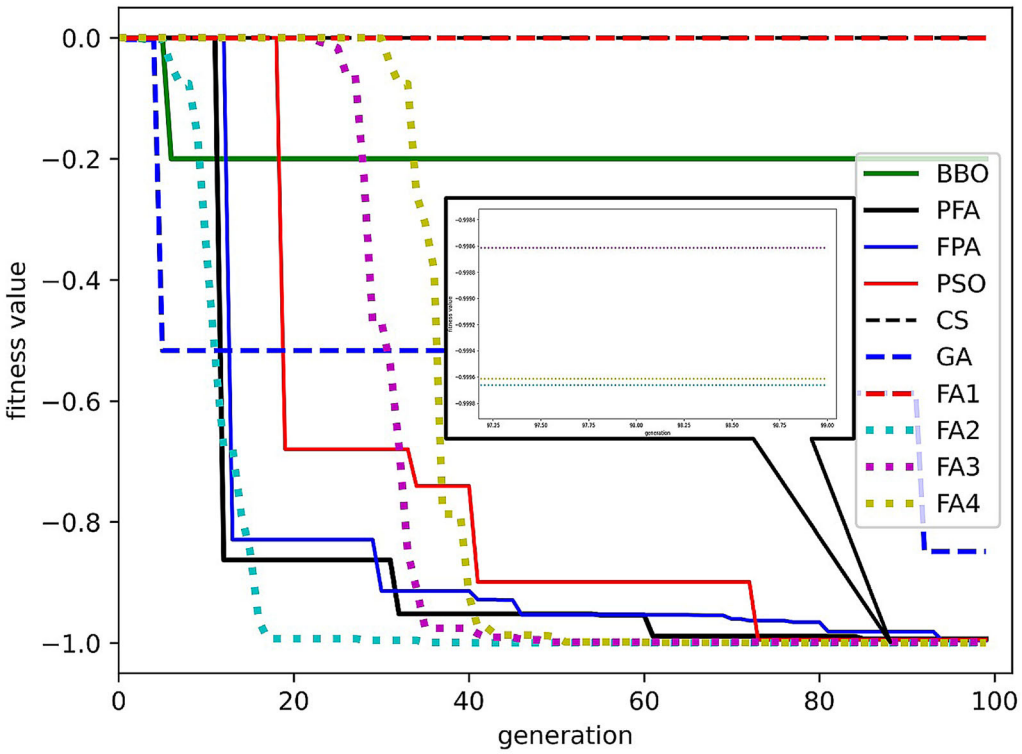

Figure 8. Performance plot of Easom Function.

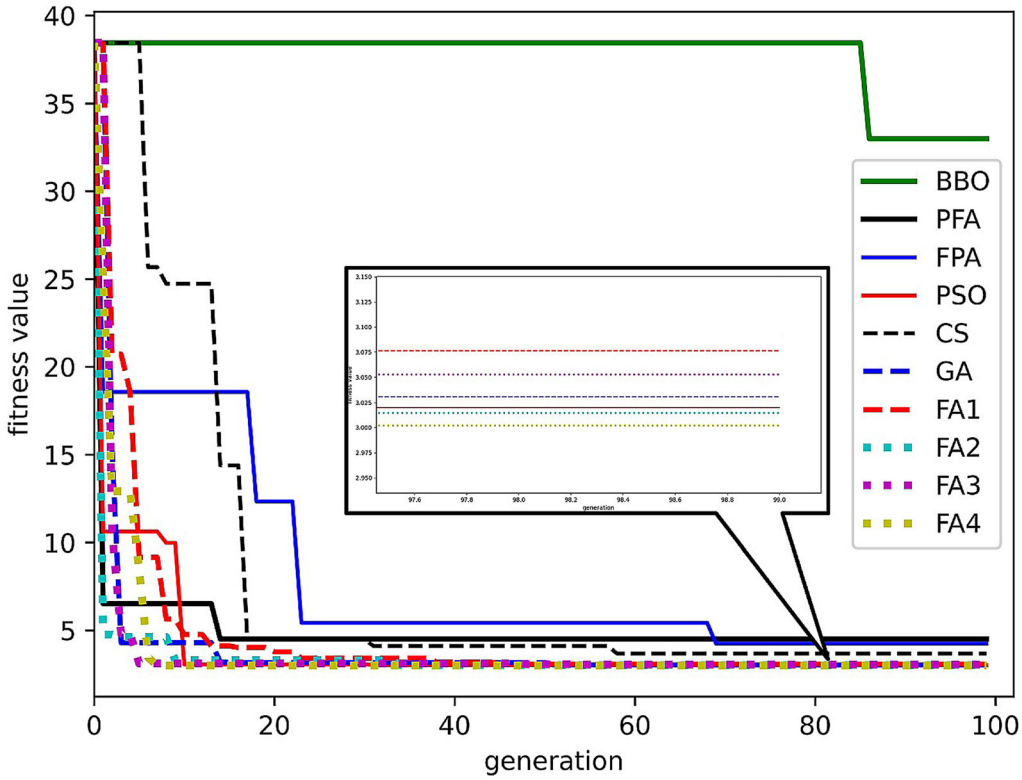

Figure 9. Performance plot of Goldstein-Price Function.

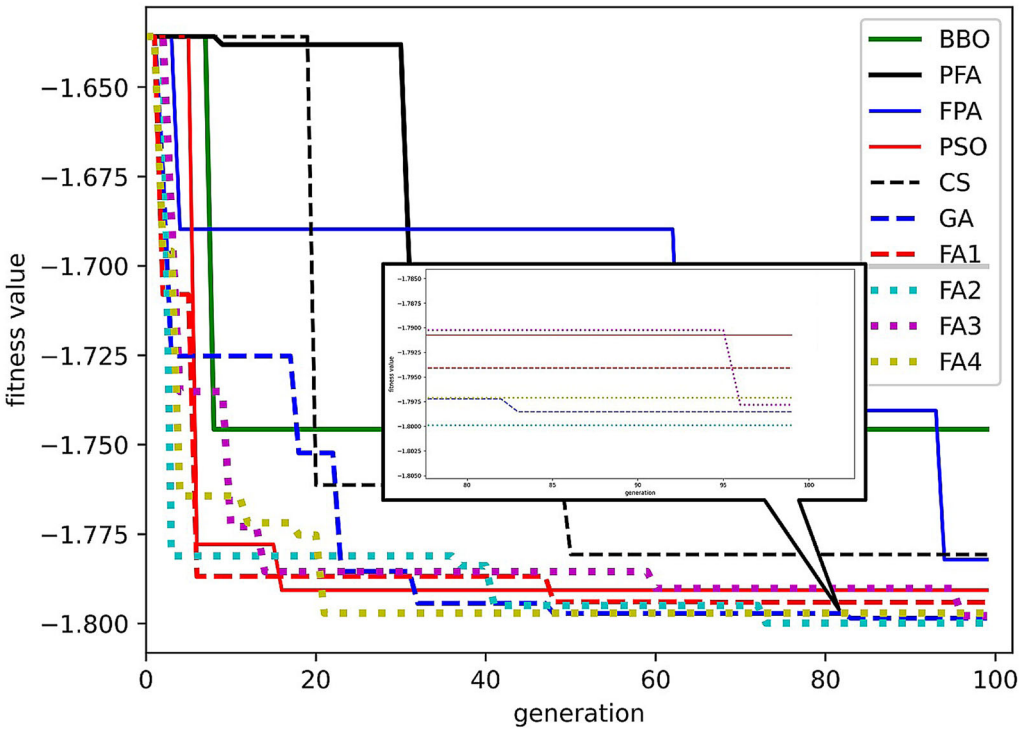

Figure 10. Performance plot of Michalewicz Function.

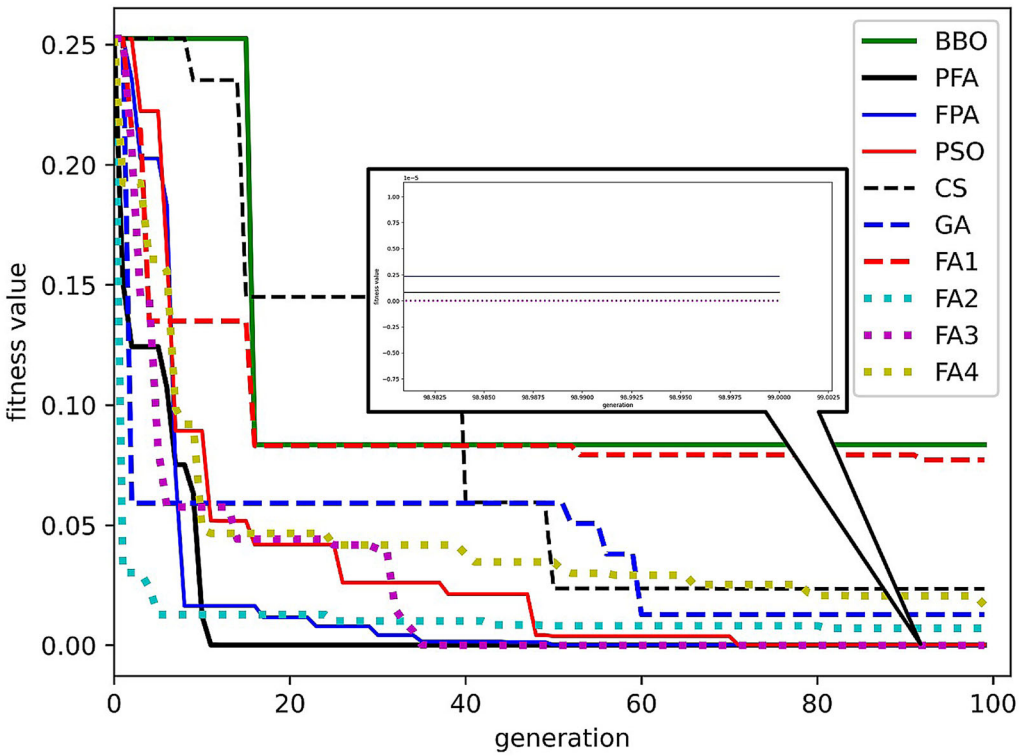

Figure 11. Performance plot of Schaffer (6) Function.

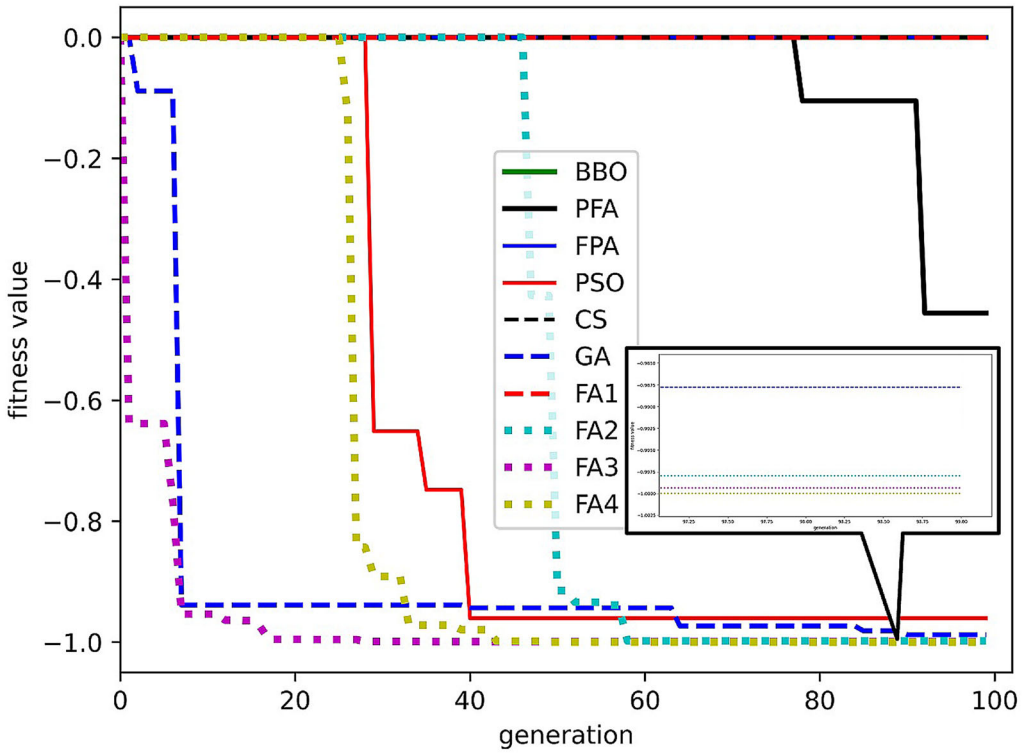

Figure 12. Performance plot of Xin-She-Yang (3) Function.

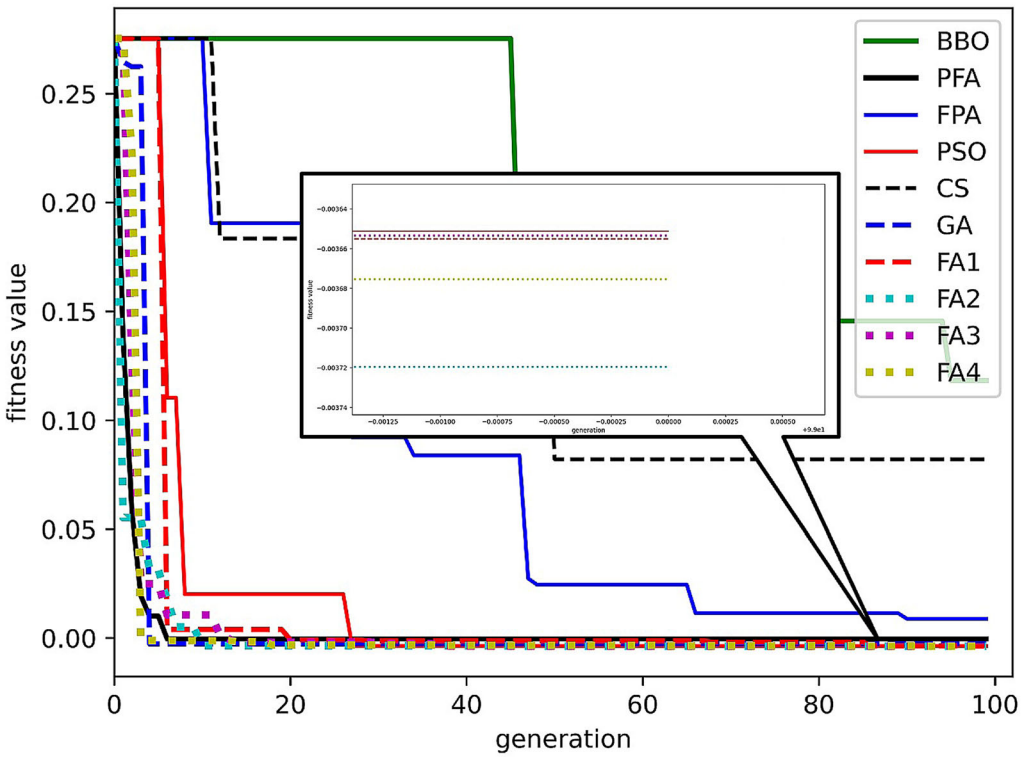

Figure 13. Performance plot of Zettl Function.

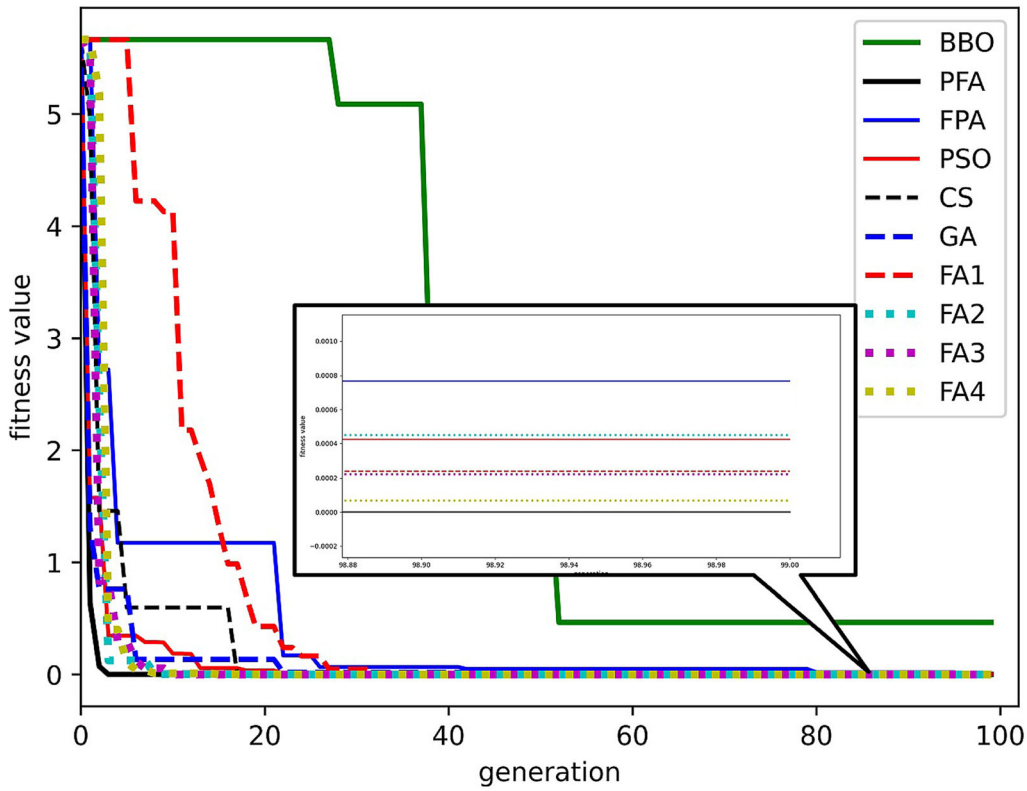

Figure 14. Performance plot of Zakharov (1) Function.

for solving a capacitated facility location problem. Sarangi *et al* [41] have modified the FA by integrating the formula used in particle swarm optimization (PSO). They have retained best positions for both individual and group in all successive iterations and have used it for the additional velocity component calculation. Aydilek *et al* [42] have enhanced hybrid FA and PSO by incorporating chaotic functions. Yelghi and Köse [43] have successfully incorporated the concept of tidal force for modifying the attraction component of the standard FA. Tomas *et al* [44] have introduced the concept of orthogonal learning to the hybridized FA and PSO algorithms to affect better performances than that obtained from the original hybridization. Huang *et al* [45] have used a hybridization of GA, FA and differential evolution (DE) to enable the maximum power point tracking for photovoltaic systems under partial shading conditions. The authors have created the hybridization framework by implementing DE mutation followed by FA attraction and finally carrying out GA crossover in each generation.

The hybrid methods of FA have been able to successfully make inroads to the hitherto unseen solution space. Nevertheless, a generic framework of integration of different independent meta-heuristics has not been standardized. There has been no such work reported which has partitioned the population and considered running different meta-heuristic algorithms in sub-groups.

1.3 *Need and novelty of new algorithm*

There have been numerous efforts for improving the standard FA to solve multiple optimization problems. At the same time they have added precious knowledge base to the literature in the process of different approaches of modifications. The key aspects that can be highlighted from the rigorous literature review are as follows:

- Randomness part can be modified with controlled direction using genetic crossover and/or mutation or following other probability distributions for better exploration.
- Records of past solutions and performance avoids probable loss of eligible solutions.
- Adaptive α comes handy especially at later stages for intensive exploitation.

Giving due respect to the previous modification efforts, it has been observed that there is no formal integration with any full fledged evolutionary algorithm as such. Genetic operators in parts have been tried as an alternate or additional update strategy of the standard FA. Their application has been targeted mainly on the fittest fireflies [40] for getting even better solutions. In some cases it has been purely application specific implementation of crossover using dominant genomes [39]. The modified algorithm in such case can not be used for solving any general

Table 4. Comparison results of functions 11 - 19 for 16-D.

Function	Alpine(1) Function		**Dixon-Price Function**		**Levy Function**	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	3375.92 ± 277.93	**2**.67246 ± **0**.4	2710.46 ± 201.2	12319.92265 ± 773.9936	3965.64 ± 63.22	**1**.85192 ± **0**.594
FPA	1815.69 ± 197.71	7.049 ± 2.23	1934.83 ± 173.86	569.36587 ± 66.568	3146.83 ± 22.28	5.72203 ± 0.22
PFA	21556.15 ± 812.64	14.79595 ± 2.975	16823.13 ± 990.31	2775.48073 ± 52.6423	19520.15 ± 1518.71	33.439 ± 9.464
CS	2313.93 ± 173.47	20.607 ± 2.054	1545.87 ± 251	58160.20214 ± 19.3558	1965.75 ± 98.73	47.29869 ± 4.894
PSO	1092.85 ± 207.2	3.757 ± 1.028	1361.88 ± 174.37	41.10107 ± 7.1484	2567.34 ± 12.92	3.6567 ± 1.412
GA	7601.56 ± 219.07	9.01469 ± 0.705	3646.29 ± 427	3315.55801 ± 51.5104	4803.67 ± 183.25	8.50226 ± 2.735
FA1	27942.47 ± 1245.24	9.09556 ± 0.633	27016.47 ± 796.63	151.40032 ± 2.7784	25364 ± 251.22	26.50476 ± 1.627
FA2	7992.92 ± 458.72	2.8544 ± 0.364	5208.79 ± 363.31	3.95043 ± 0.6344	6354.28 ± 166.73	5.15981 ± 0.992
FA3	11159.64 ± 760.51	**2**.8309 ± **0**.49	8778.96 ± 316.25	**1**.86466 ± **0**.2025	9882.08 ± 28.4	2.46923 ± 1.519
FA4	16371.42 ± 1133.56	**2**.29278 ± **0**.365	13716.4 ± 442.46	**0**.90298 ± **0**.2512	14844.66 ± 279.68	**1**.71954 ± **2**.47

Function	Rastrigin Function		Rosenbrock Function		Sphere Function	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	2003.15 ± 102.26	**22**.17343 ± **1**.168	2649.9 ± 52.16	76339.53829 ± 704.733	2651 ± 150.42	5.69212 ± 1.272
FPA	1704.93 ± 69.84	**69**.17319 ± **8**.529	1713.42 ± 2.82	2599.38309 ± 16.788	1546.37 ± 214.18	19.48063 ± 25.829
PFA	12671.67 ± 385.77	139.28111 ± 28.208	16624.32 ± 13.96	1687225.534 ± 666.174	16599.76 ± 701.42	238.55139 ± 29.621
CS	1094.56 ± 0.71	156.65561 ± 8.411	1598.21 ± 6.33	15738070.75 ± 1242.74	1489.73 ± 238.6	172.67592 ± 1.511
PSO	798.37 ± 2.12	64.39563 ± 9.968	1066.15 ± 2.82	46137.90181 ± 458.758	882.64 ± 41.03	6.50556 ± 16.944
GA	2734.68 ± 21.18	88.90463 ± 8.463	3624.83 ± 7.74	808516.8752 ± 206.606	3373.93 ± 383.5	134.05013 ± 10.504
FA1	21703.01 ± 22.54	146.47182 ± 4.865	25269.4 ± 316.43	4286885.836 ± 1240.917	25111.61 ± 996.93	0.29449 ± 0.087
FA2	3850.19 ± 37.36	90.45436 ± 2.364	5469.7 ± 141.5	**147**.86933 ± **15**.612	4931.06 ± 412.44	0.08064 ± 0.027
FA3	6799.33 ± 35.96	79.58992 ± 13.137	9792.43 ± 34.88	**71**.18291 ± **14**.925	8444.63 ± 246.77	**0**.05245 ± **0**.013
FA4	11328.21 ± 2.1	73.43202 ± 8.385	14960.71 ± 587.85	184.92492 ± 15.292	13322.69 ± 969.73	**0**.02323 ± **0**.01

Function	Step(2) Function		Sum Squares Function		Xin-She-Yang (1) Function	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	2617.41 ± 40.87	414.3257 ± 147.078	2418.02 ± 31.71	7068.28929 ± 753.904	2780.87 ± 32.88	0.21958 ± 0.169
FPA	1830.11 ± 11.28	679.5015 ± 2.121	1577.99 ± 76.4	2951.582 ± 7.114	1651.02 ± 6.26	9.20017 ± 3.088
PFA	16610.34 ± 12.37	1452.7321 ± 75.433	16652 ± 74.21	14618.92605 ± 456.184	16512.81 ± 66.09	3.72492 ± 0.995
CS	1561.32 ± 4.94	14652.5124 ± 454.67	1526.63 ± 3.23	114105.2836 ± 948.057	1670.54 ± 135.4	2874.54363 ± 195.053
PSO	925.03 ± 43.02	389.3239 ± 18.019	846.24 ± 47.25	**1813**.1691 ± **19**.58	987.36 ± 1.41	1.09523 ± 0.698
GA	3419.36 ± 21.86	3369.5217 ± 526.087	3371.63 ± 34.9	21197.44893 ± 80.87	3484.34 ± 1.71	3.29214 ± 0.486
FA1	24018.71 ± 49.78	22734.5293 ± 101.116	23959.46 ± 146.95	151356.0888 ± 170.783	26223.42 ± 972.43	0.0892 ± 0.021
FA2	4917.85 ± 18.33	342.5792 ± 15.685	4954.64 ± 54.24	2145.34541 ± 227.942	5301.37 ± 188.24	0.01764 ± 0.014
FA3	8482.68 ± 40.58	**325**.5792 ± **26**.163	8501.71 ± 42.46	2533.63824 ± 103.309	8824.72 ± 45.72	**0**.01058 ± **0**.007
FA4	13360.51 ± 137.22	**206**.1934 ± **14**.664	13590.18 ± 39.29	**1709**.19545 ± **141**.555	13845.02 ± 111.97	**0**.00921 ± **0**.004

optimization problems. If in any scenario it has been used to evolve the weaker solutions alone, additional evaluations of objective function fitness have been performed incurring more computational cost.

1.4 *Motivation and inspiration*

A single method of modification may not capture all the needed improvements of the existing FA. This may raise issues with premature convergence in problems with higher dimensionality. A very handful cases have concentrated on combining different approaches of modifications. A very few of the previous works reported has targeted hybridization of different meta-heuristics in true sense. Hence, this work aims to capitalize on the strengths of two algorithms viz. faster convergence of GA and better accuracy of FA so as to achieve a better performing algorithm.

In this approach, a novel methodology is formulated in the form of partition cum unification based genetic - FA

Table 5. Comparison results of functions 11 - 19 for 32-D.

Function Name	Alpine(1) $f^* = 0$		Dixon-Price $f^* = 0$		Levy $f^* = 0$	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	4151.92 ± 37.24	9.46719 ± 1.849	4253.91 ± 94.01	18229.19278 ± 583.1756	6042.92 ± 40.15	**13**.31744 ± **2**.189
FPA	4174.44 ± 4.25	23.21317 ± 0.84	4105.64 ± 8.46	16302.97239 ± 805.55	5826.54 ± 6.35	40.47595 ± 1.444
PFA	22736.56 ± 141.61	47.91859 ± 0.226	22572.46 ± 116.6	122989.6484 ± 642.1208	24183.25 ± 1.01	119.25448 ± 61.335
CS	3957.55 ± 51.5	47.90387 ± 3.311	3968.01 ± 45.14	682456.0873 ± 715.2843	4367.58 ± 2.32	131.41598 ± 36.709
PSO	3372.6 ± 9.87	16.17078 ± 4.132	3331.25 ± 3.6	7687.38604 ± 189.2988	5004.24 ± 7.05	13.59586 ± 2.082
GA	8039.94 ± 25.41	24.19729 ± 1.644	6529.69 ± 3.49	31071.63456 ± 512.5961	8163.85 ± 5.61	39.20025 ± 3.582
FA1	34290.83 ± 174.78	28.21906 ± 2.14	34023.49 ± 68.77	27738.38929 ± 115.0443	35727.5 ± 50.1	66.65928 ± 2.715
FA2	8433.11 ± 2.77	**8**.91902 ± **1**.227	8408.64 ± 7.75	**95**.98027 ± **12**.0397	10072.69 ± 7.06	14.30951 ± 0.74
FA3	13131.59 ± 29.56	11.79566 ± 1.541	13068.22 ± 2.02	**64**.14974 ± **2**.8246	14736.75 ± 19.04	14.68524 ± 2.047
FA4	20260.08 ± 45.84	**8**.62356 ± **0**.44	20581.14 ± 244.81	107.35544 ± 22.0462	21768.71 ± 53.27	**11**.45863 ± **4**.255

Function Name	Rastrigin $f^* = 0$		Rosenbrock $f^* = 0$		Sphere $f^* = 0$	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	4445.28 ± 48.6	**88**.14888 ± **9**.641	4197.97 ± 5.03	664508.5654 ± 330.784	4131.19 ± 77.03	29.32855 ± 2.884
FPA	4210.89 ± 69.78	**190**.8849 ± **25**.989	4070.73 ± 8.46	171261.4486 ± 542.892	3850.85 ± 0.62	109.15806 ± 7.632
PFA	22602.22 ± 27.97	350.61797 ± 33.617	23133.12 ± 29.65	22211614.02 ± 820.138	22456.49 ± 20.61	740.59813 ± 9.326
CS	3956.52 ± 4.92	391.51587 ± 13.903	3952.55 ± 23.27	98607940.06 ± 235.38	3849.81 ± 61.34	549.39435 ± 16.107
PSO	3361.14 ± 12.01	220.39739 ± 6.072	3332.71 ± 1.41	321449.9769 ± 187.004	3029.55 ± 0.05	31.50217 ± 9.422
GA	6584.04 ± 5.64	232.0219 ± 18.538	6543.21 ± 0.11	7467747.463 ± 653.833	6309.8 ± 8.51	419.08603 ± 33.061
FA1	34510.81 ± 59	365.82424 ± 4.558	34117.03 ± 41.64	42864576.94 ± 918.677	36270.24 ± 117.62	32.0092 ± 0.894
FA2	8473.99 ± 2.78	233.55792 ± 3.983	8389.23 ± 0.04	1944.16522 ± 114.53	8193.22 ± 62.77	**1**.49072 ± **0**.454
FA3	13192.4 ± 70.48	230.69942 ± 1.357	13122.57 ± 55.09	**1160**.32527 ± **341**.623	12919.18 ± 86.13	**1**.2801 ± **0**.09
FA4	20633.02 ± 193.24	214.02865 ± 17.308	20196.48 ± 21.2	**1589**.57498 ± **439**.686	20032.62 ± 75	1.89594 ± 0.225

Function Name	Step(2) $f^* = 0$		Sum Squares $f^* = 0$		Xin-She-Yang (1) $f^* = 0$	
Algorithm	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value	Mean CPU Time (ms)	Mean Objective Function Value
BBO	4662.37 ± 318.79	**3320**.21 ± **593**.97	3995.25 ± 161.47	161352.35 ± 169.373	4359.12 ± 139.77	79.61591 ± 1.22689
FPA	4520.93 ± 255.49	**856**.13 ± **395**.98	3789.17 ± 27.16	169186.25 ± 185.183	4210.94 ± 232.58	452589.2685 ± 323.566
PFA	25346.92 ± 1656.09	13146.19 ± 759.433	23133.12 ± 29.65	196667.4749 ± 402.781	23876.65 ± 184.61	2992457.495 ± 402.994
CS	4378.09 ± 56.55	45050.51 ± 835.938	14630.33 ± 598.61	717051.7199 ± 487.797	4055.18 ± 2.68	27575755784.34 ± 3506.12
PSO	3360.58 ± 24.57	1693.13 ± 442.649	3008.82 ± 22.2	189540.70 ± 822.851	3299.62 ± 41.59	2.48656 ± 1.696
GA	8026.8 ± 579.49	12083.32 ± 415.78	6861.35 ± 31.86	158967.17 ± 823.89	6801.15 ± 60.05	9205.93 ± 121.67
FA1	38597.426 ± 708.22	65868.710 ± 10.343	36921.770 ± 58.16	1026759.599 ± 947.17	39955.330 ± 62.95	41480.496 ± 541.498
FA2	8580.52 ± 256.18	4912.54 ± 305.47	8709.04 ± 45.23	**52836**.55 ± **443**.68	8839.84 ± 51.6	**0**.11752 ± **0**.03
FA3	13699.65 ± 585.68	6098.57 ± 379.72	14116.53 ± 149.47	**57148**.79 ± **172**.88	13724.66 ± 704.44	0.98629 ± 1.27
FA4	21780.92 ± 574.49	5319.51 ± 149.117	21947.96 ± 325.32	71005.68905 ± 261.071	21495 ± 32.79	**0**.08405 ± **0**.109

(PUBG-FA) by integrating FA with GA. The initial population is partitioned into two compartments based on a weight factor, w set by user, which determines how much of the algorithm is dominated by FA. The first compartment which has the superior fireflies based on light intensity (fitness) runs a little improved version of standard FA, whereas the second chamber consisting of the weaker solutions runs the entire GA encompassing selection, crossover and mutation. After each iteration these two populations are again unified and the intensities of the new fireflies are computed. Based on that, the consolidated population is sorted and trimmed before going to the next iterative cycle for further partition, update and unification. The new algorithm in three variants of different w (30%, 50% and 70%) is tested and compared with the two constituents i.e. standard firefly and genetic algorithm and additionally with some state-of-the-art meta-heuristics namely PSO [46], cuckoo search (CS) [47, 48], flower pollination algorithm (FPA) [49], pathfinder algorithm (PFA) [50], bio-geography based optimization (BBO) [51]

on 19 benchmark objective functions of different complexity, continuity, differentiability, mode, separability and scalability. This testing is carried-out considering several dimensionality of the problems viz. 2-D, 16-D, and 32-D. Further, the new algorithm is tested and compared with others on two real engineering optimization problems namely tension-compression spring [52, 53] and 3-bar truss problem [53, 54]. Non-parametric statistical tests, namely Wilcoxon rank-sum tests are conducted on the objective function values for each dimensional (scalability) test for all the algorithms from repeated independent experiments to statistically rule-out any significant deviations. One of the extensively used multi-criteria decision making (MCDM) tools, namely, technique for order of preference by similarity to ideal solution (TOPSIS) [55] is applied to statistically determine the best performing algorithm given the different multi-dimensional test scenarios and two assessment criteria - running time and objective function value.

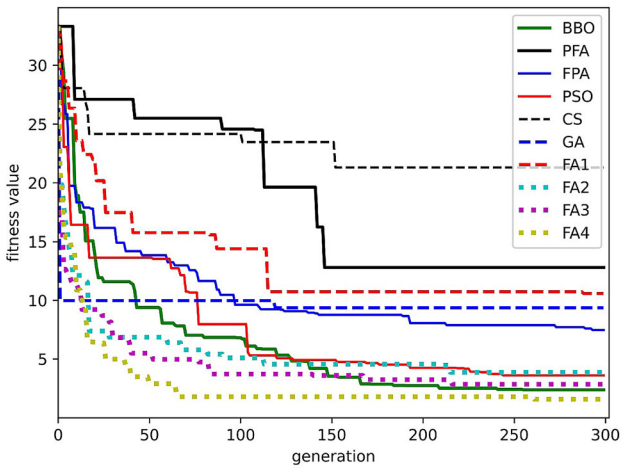

Figure 15. Performance plot of Alpine(1) Function for 16-D.

Thus, the primal novel prospects and contributions of this article can be summarized as follows:

- Seeking key improvement areas of FA from the past literature and combining them together for getting added advantages.
- Envisaging a genetic strategy so as to achieve a faster convergence rate than conventional FA.
- Partitioning the population for running two algorithms in two compartments and unification of the population derived from respective compartments.

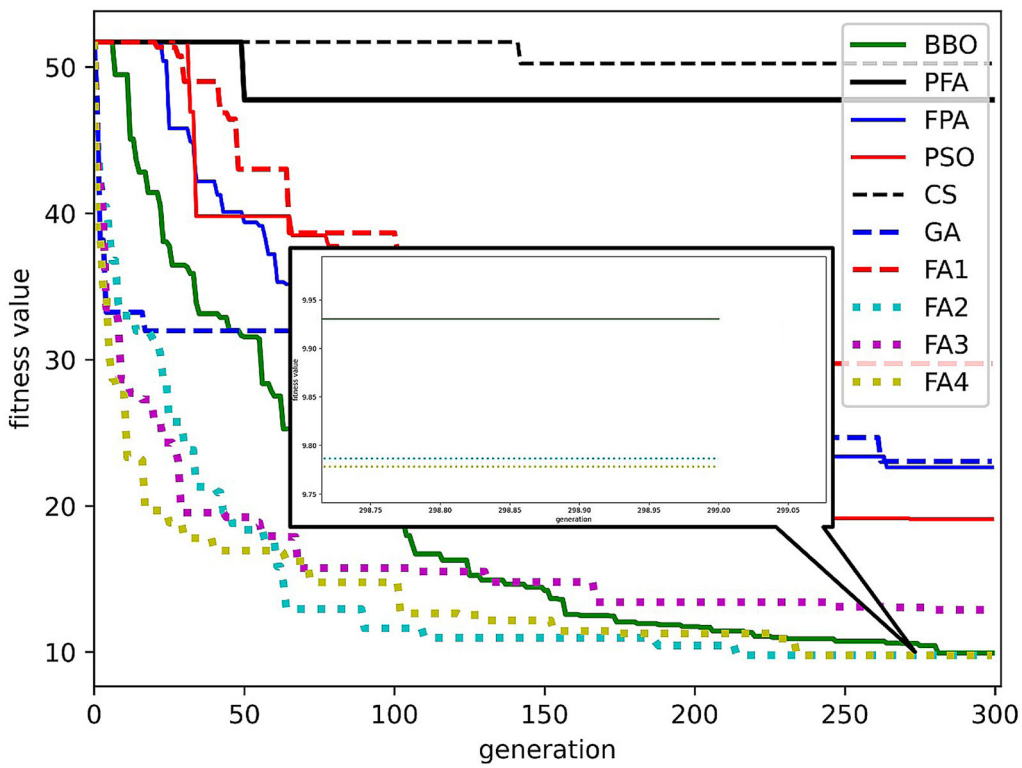

Figure 16. Performance plot of Alpine(1) Function for 32-D.

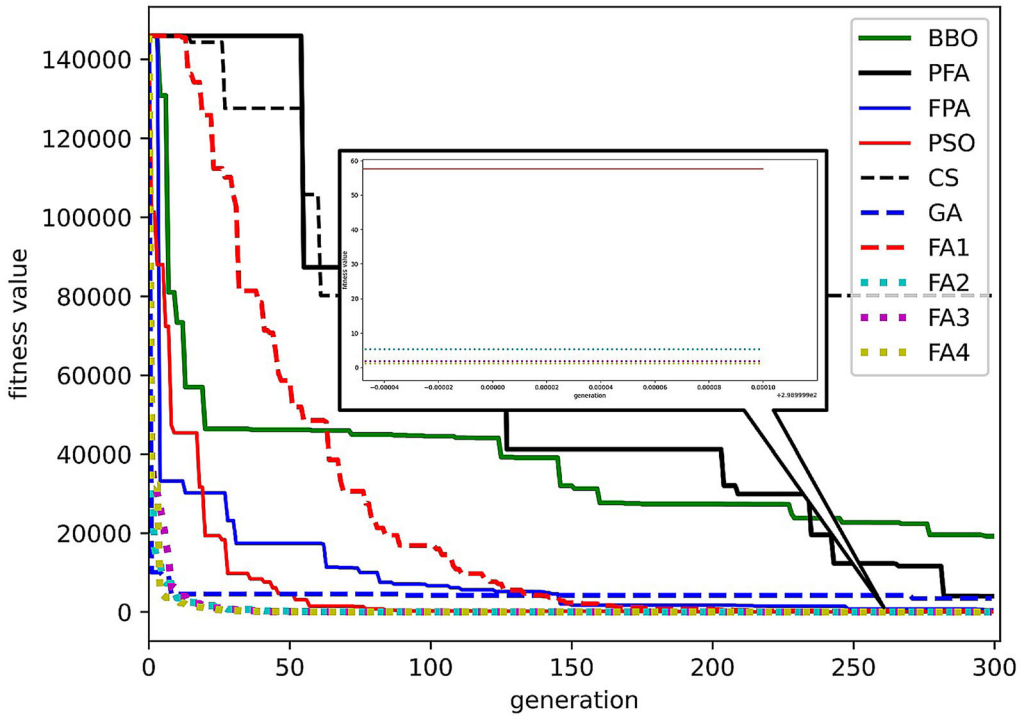

Figure 17. Performance plot of Dixon-Price Function for 16-D.

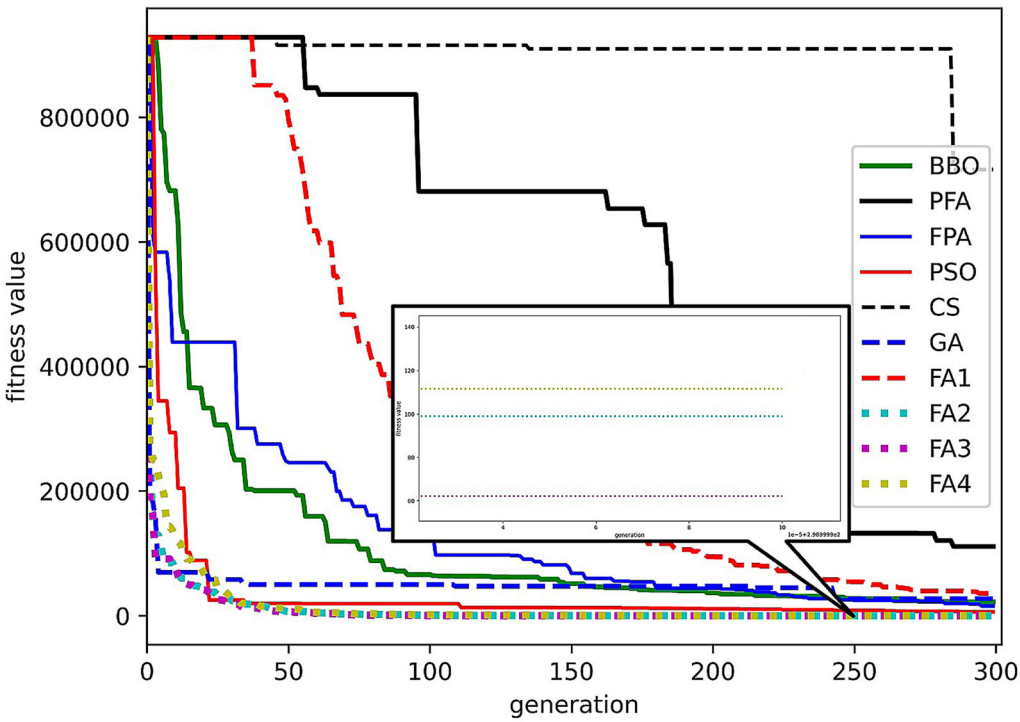

Figure 18. Performance plot of Dixon-Price Function for 32-D.

- Keeping the algorithmic parameter to tune to a single weight factor indicating the dominance of one algorithm over another.
- Utilizing MCDM tool to strategically decide the final ranks of the competing optimization algorithms.

The rest of the paper is segmented as follows: section 2, where the methodology is described, presents the new algorithm elaborately and describes the objective functions and engineering optimization problems considered for testing. Results of all the testing are discussed in section 3. Finally, some concluding remarks are drawn in section 4.

2. Methodology

The basic idea of the new algorithm PUBG-FA is pretty simple. Dimmer fireflies get attracted towards the brighter ones and by virtue of their positions getting updated, their light intensities or objective fitness values are improved. The weaker section having feeble flashing intensity, instead of waiting for their turn for improving up the ladder of intensity through generations, are moved into a separate genetic compartment. In this space they get opportunities for further and faster improvement through explorations by applying standard genetic operators, namely selection, crossover and mutation. The remaining best part of the population having brighter fireflies undergo a little improved version of the standard FA in the primary compartment. Memory of the candidate solutions, random movement of the brightest firefly and adaptive α constitute the improvement spectrum of FA as already successfully experimented in some previously reported works [19, 20, 36]. Figure 1 shows the flowchart of the new algorithm. After each iteration, the populations from two compartments along with the past memory are unified and ranked according to the light intensity meaning the objective fitness value before venturing for the next cycle. In the pseudo code presented in Algorithm 1, the new algorithm can be easily comprehended. Key improvement areas and features of PUBG-FA are discussed in the following subsection.

2.1 *Features of PUBG-FA*

2.1a *Memory of past records*: In the standard FA, the positions of all the fireflies are updated in a stochastic goal direction. Naturally, it may so happen that the intensity of the light meaning the objective fitness deteriorates from the previous value. Hence, it is changed to keep memory of the past records so that eligible candidates once found are never lost.

2.1b *Random movement of the brightest*: The position of the brightest firefly is not updated by the standard FA as because only dimmer fireflies update positions with respect to the brighter ones. Therefore, the algorithm is modified so that the brightest one is updated by the following Equation (2) involving only random term.

$$x_i = x_i + \alpha(\varepsilon() - 0.5) \tag{2}$$

Algorithm 1 PUBG-Firefly Algorithm

$pop_s = w \times pop$ ▷ (partition pop based on weight factor w set by user)
$list \leftarrow InitializeFireflies(pop)$ ▷ (randomize positions with (-)INF intensity)
$It \leftarrow 1$ ▷ (counter for generation/iteration)
for $It \leq MaxGen$ **do** ▷ ($MaxGen$ is the termination criteria)
  $[list_s, list_i] \leftarrow Partition(list, pop_s)$ ▷ (partition population into two compartments)
  ▷ (***Start of Firefly Compartment***)
  $list_c \leftarrow CloneFirefly(list)$ ▷ (preserve the history of fireflies)
  for $(i \leftarrow 2) \rightarrow pop_s$ **do**
    for $(j \leftarrow 1) \rightarrow i$ **do**
      if $GetIntensity(j) > GetIntensity(i)$ **then**
        $list_s[i] \leftarrow UpdatePosition(i)$ ▷ (update by equation 1)
      end if
    end for ▷ (inner for loop j)
    $ComputeIntensity(list_s[i])$ ▷ (compute new intensity after updates)
  end for ▷ (outer for loop i)
  $RandomizeBrightest()$ ▷ (randomize the brightest using equation 2)
  $ComputeBrightestIntensity()$
  ▷ (***End of Firefly & Start of Genetic Compartment***)
  $list_i \leftarrow GeneticSelection(list_i)$ ▷ (RouletteWheel selection)
  $list_i \leftarrow GeneticCrossover(list_i)$ ▷ (crossover by equation 4 and 5)
  $list_i \leftarrow GeneticMutation(list_i)$ ▷ (mutation by equation 6)
  ▷ (***End of Genetic Compartment***)
  $list_u \leftarrow list_c + list_s + list_i$ ▷ (unification of compartments)
  $list_u \leftarrow RankFireflies(list_u)$ ▷ (rank fireflies based on intensity)
  $list \leftarrow TrimFirefliesSize(list_u, pop)$ ▷ (trim the fireflies for pop size)
  $\alpha \leftarrow \alpha \times 0.95$ ▷ (equation 3)
end for ▷ (generation counter loop)
▷ (**Return the best solution and visualize**)

2.1c *Adaptive random coefficient*: It has been realized by the past researchers that an adaptive α improves the exploration in the beginning and exploitation in the final

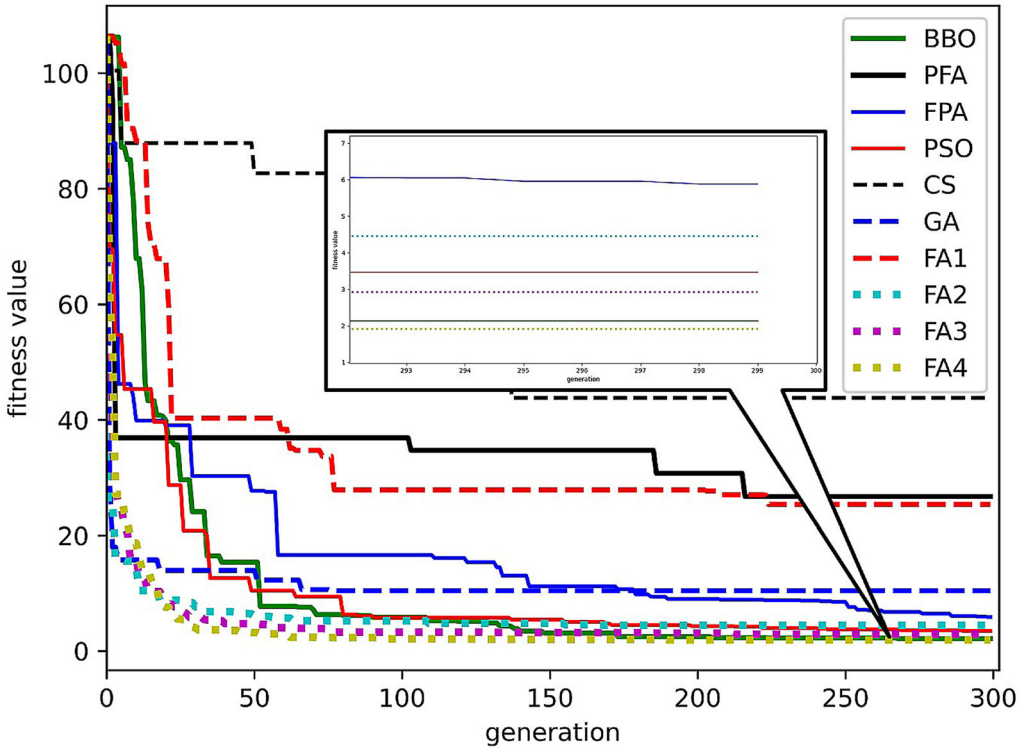

Figure 19. Performance plot of Levy Function for 16-D.

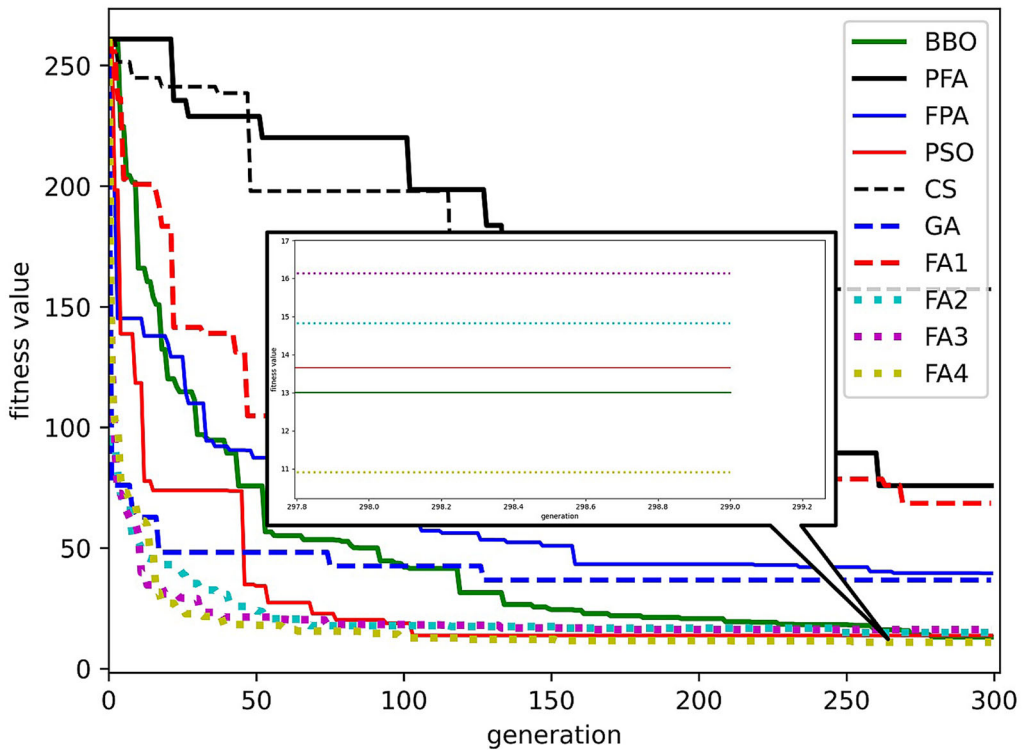

Figure 20. Performance plot of Levy Function for 32-D.

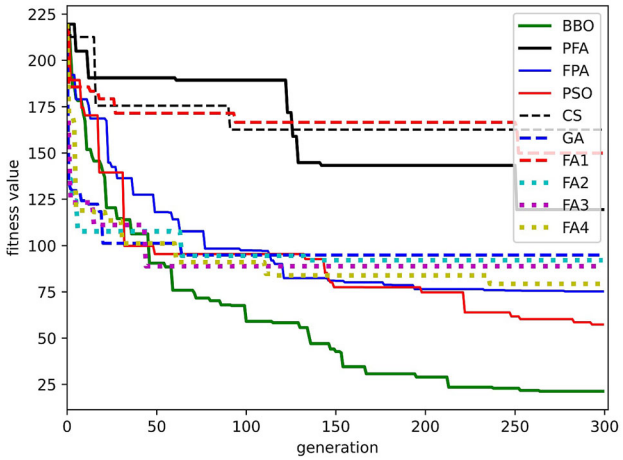

Figure 21. Performance plot of Rastrigin Function for 16-D.

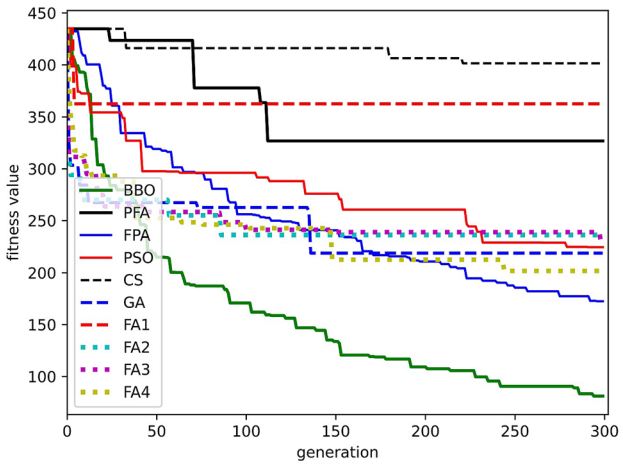

Figure 22. Performance plot of Rastrigin Function for 32-D.

stages of the FA. The following Equation (3) is used to update α based on the value of initial randomness coefficient α_0 value and current generation/iteration counter *iter*.

$$\alpha = \alpha_0 \times 0.95^{iter} \tag{3}$$

2.1d *Partitioning the population*: The initial population of size *pop* with randomly assigned positions is partitioned into two compartments based on *w*, which actually says $w \times pop$ would go into FA compartment and the rest i.e. $(1 - w) \times pop$ would go into the GA counterpart. This essentially makes the number of total functional evaluations same as that of the standard FA while improving the exploration capability by applying integrated GA.

2.1e *Genetic operators*: The specific implementations of genetic operators immensely influence the performance of GA. In this real coded GA, RouletteWheel selection mechanism is adopted. Uniform crossover as shown in Equations (4) and (5), is implemented. with a crossover probability of 0.8. Mutation operator is implemented by taking average of a random fraction $\varepsilon()$ in [0, 1] and previous value on the same dimension of position vector with a mutation probability of 0.2 as presented in Equation (6).

$$X_{child1} = X_{parent1} \times \varepsilon() + X_{parent2} \times (1 - \varepsilon()) \tag{4}$$

$$X_{child2} = X_{parent1} \times (1 - \varepsilon()) + X_{parent2} \times \varepsilon() \tag{5}$$

$$X_{mutated} = \frac{1}{2} \times (X_{child} + \varepsilon()) \tag{6}$$

2.1f *Unification before ranking*: The position-updated populations from the two compartments are unified and ranked according to their newly computed light intensities or fitness values. This approach is better than parallel processing of FA and GA in two compartments without mixing. This is because the current scheme i.e. unification creates more chances to the newly evolved fireflies from GA compartment to come to the mainstream FA and thereby converge rapidly by following a local direction of search.

2.2 *Objective functions for testing*

The new algorithm in three variants with 30%, 50% and 70% of *w* is tested on 19 benchmark functions [56, 57] as illustrated in table 1 and the results are compared with that of the standard FA and GA and some five state-of-the-art meta-heuristic algorithms, namely PSO, CS, FPA, PFA, and BBO. Here, the different variants of this algorithm is used to study the impact of *w* on the overall performance. In this study, three main indices used for comparison are mean CPU time, mean number of functional evaluations and mean value of objective function presented by the best solution. The population size and maximum generation for termination for the 2-D functions (function 1 to 10 in table 1) are kept at 50 and 100 respectively. The respective values are chosen as 200 and 300 for all 16-D and 32-D functions (function 11 to 19 in table 1) for the enhanced complexity of the problems. The number of independent repeated trials is set at 30 for all the functions. The parameter settings for all the algorithms are fixed as per the values in table 2. To make the comparative study really unbiased, all the algorithms are started with the same initial random population set.

2.3 *Engineering case study problems for testing*

The algorithms have been also tested on some real world, engineering case study problems which are described in the following sub-sections:

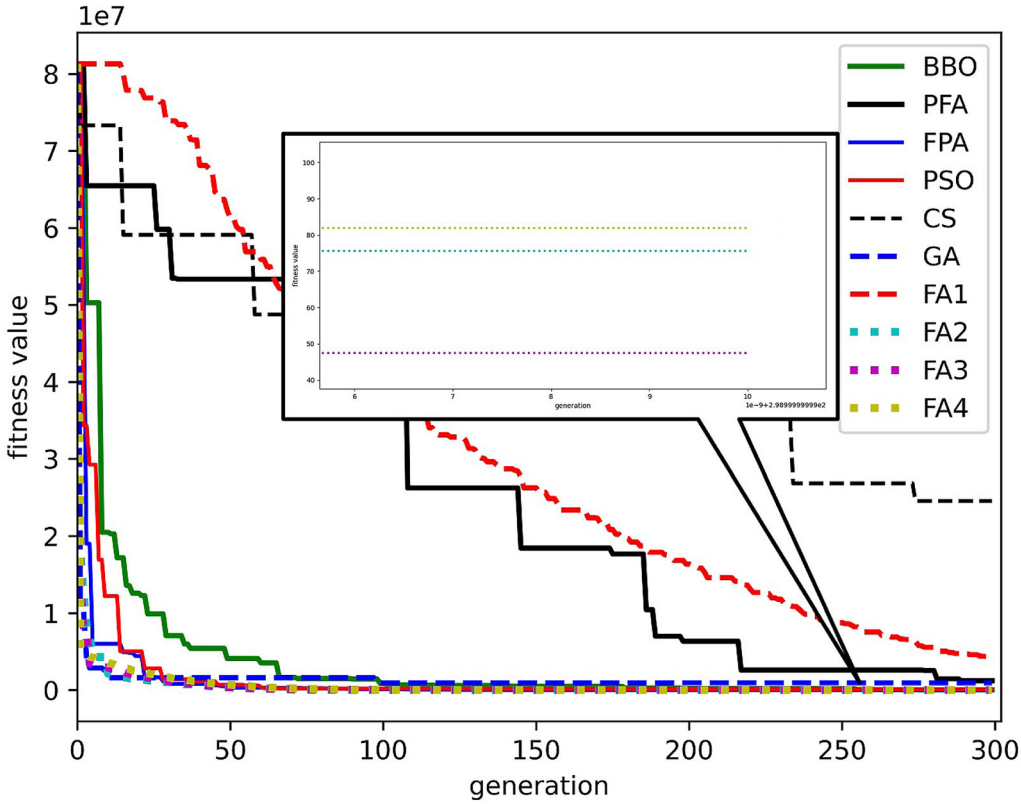

Figure 23. Performance plot of Rosenbrock Function for 16-D.

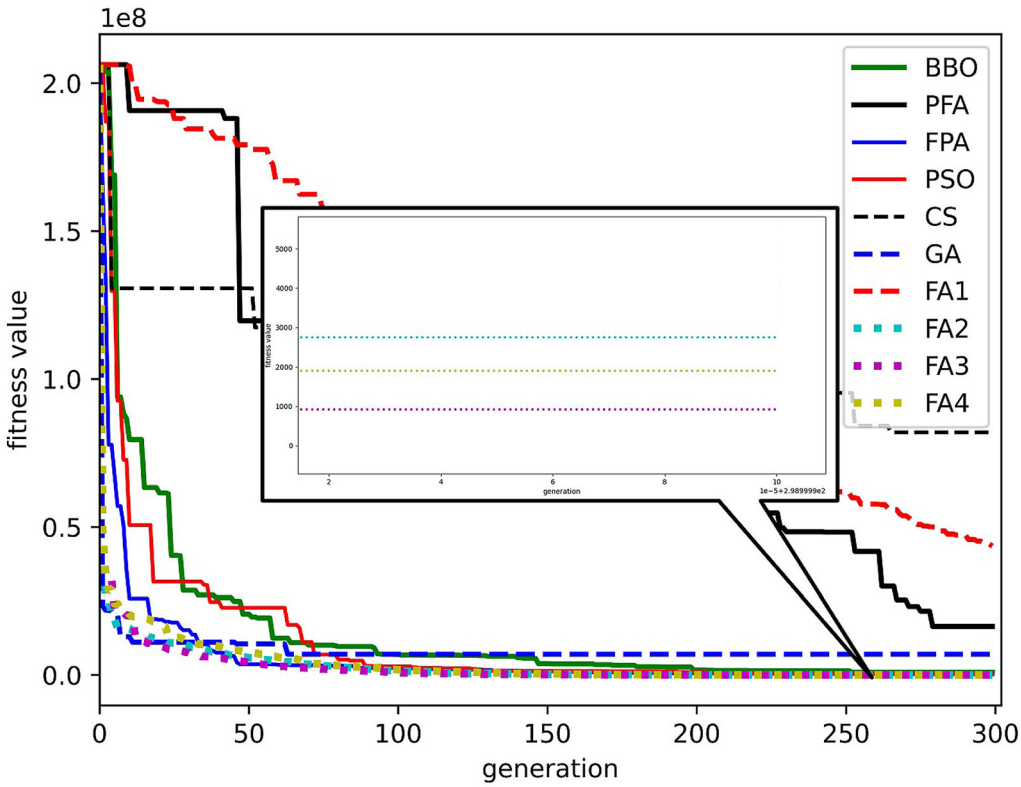

Figure 24. Performance plot of Rosenbrock Function for 32-D.

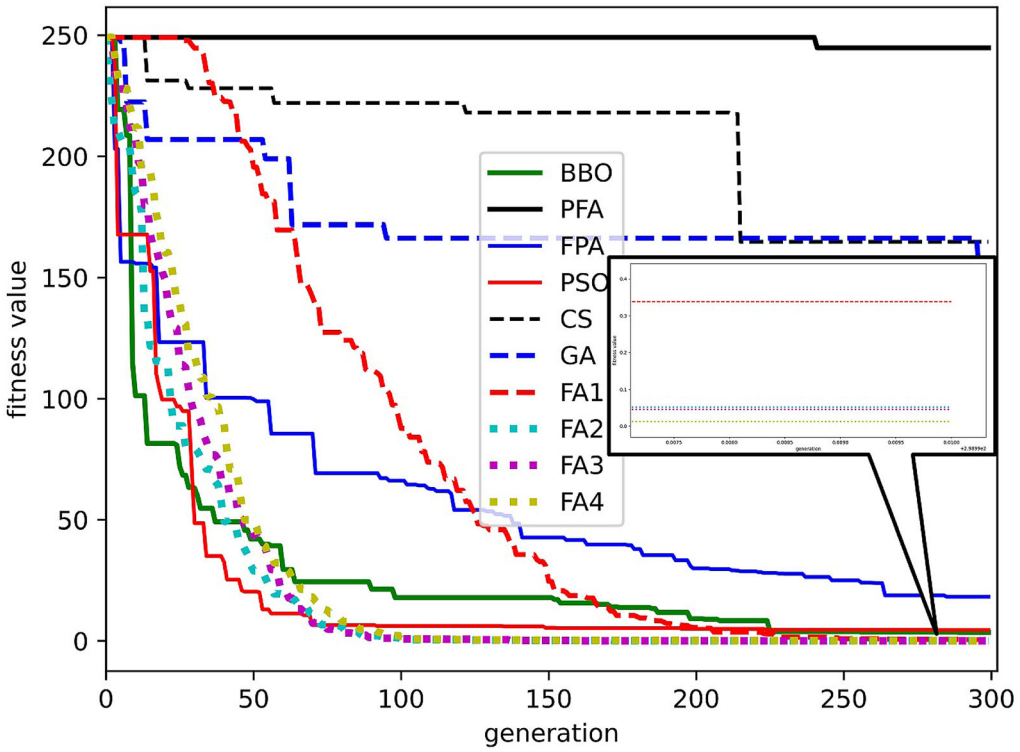

Figure 25. Performance plot of Sphere Function for 16-D.

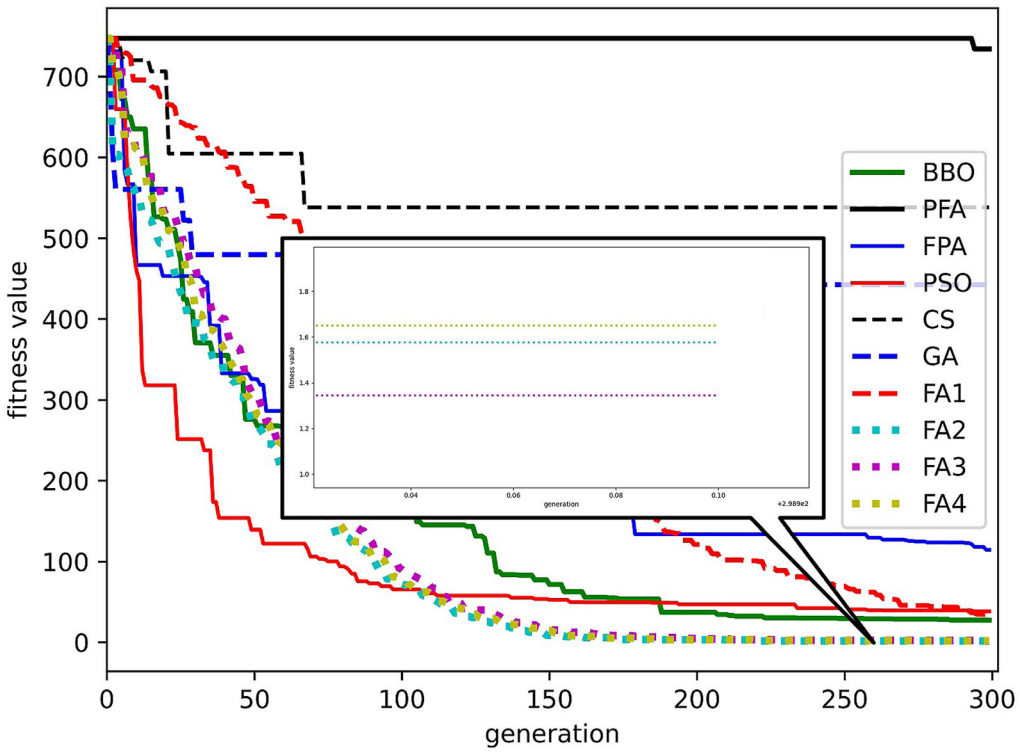

Figure 26. Performance plot of Sphere Function for 32-D.

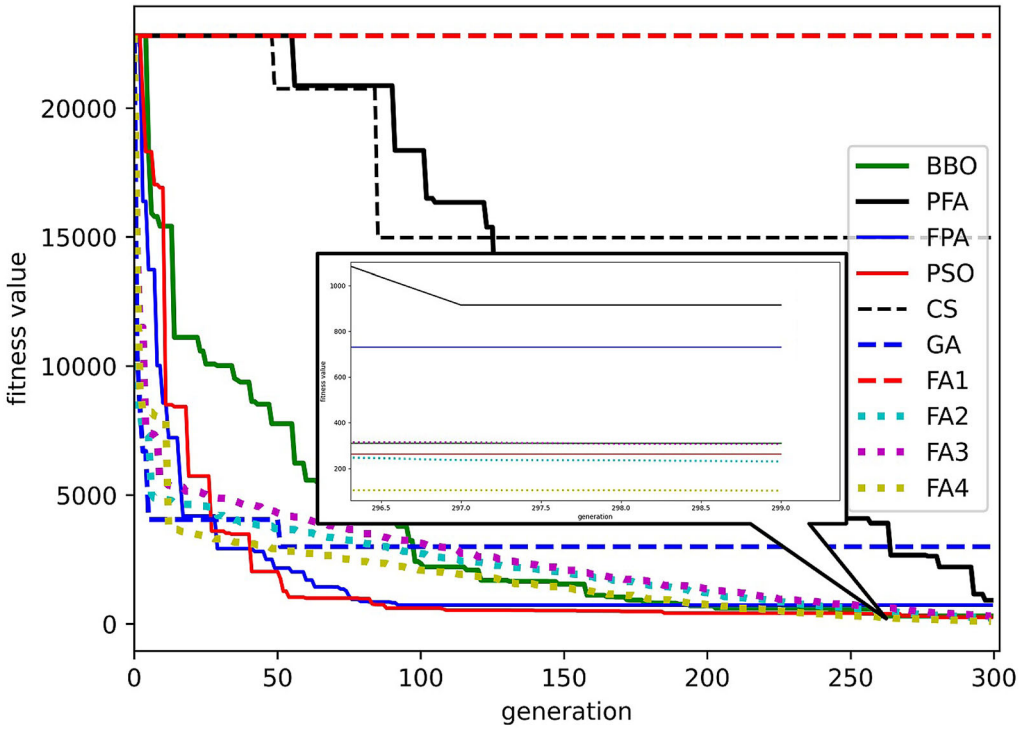

Figure 27. Performance plot of Step (2) Function for 16-D.

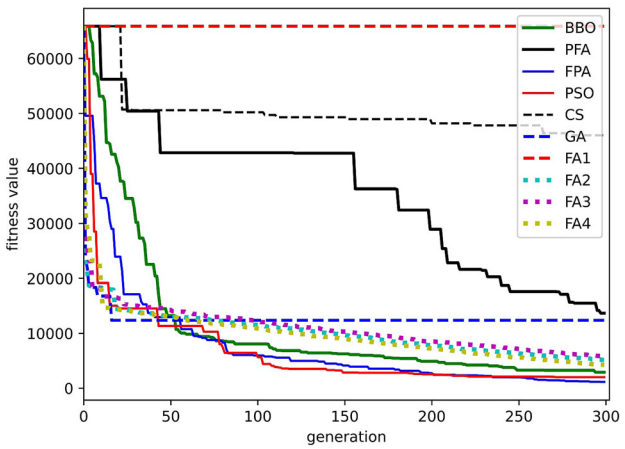

Figure 28. Performance plot of Step (2) Function for 32-D.

3.1a *Design optimization of tension/compression helical spring*: In this case, the objective is to minimize the weight of a helical spring under tension/compression. Design variables for this case study are wire diameter (d), mean coil diameter (D), and the number of active coils (N) (figure 2). Here, the constraints on shear stress, surge frequency, and minimum deflection should be satisfied during the weight optimization. The objective function and the constraints of this problem can be formulated as follows:

$$\begin{aligned}
&\textit{Consider } \vec{x} = [x_1 x_2 x_3] = [dDN],\\
&\textit{Minimize } f(\vec{x}) = (x_3 + 2) x_2 {x_1}^2,\\
&\textit{Subject to}\\
&g_1(\vec{x}) = 1 - \frac{{x_2}^3 x_3}{71785 {x_1}^4} \leq 0,\\
&g_2(\vec{x}) = \frac{4{x_2}^2 - x_1 x_2}{12566({x_2 {x_1}^3 - {x_1}^4})} + \frac{1}{5108 {x_1}^2} - 1 \leq 0,\\
&g_3(\vec{x}) = 1 - \frac{140.45 x_1}{{x_2}^2 x_3} \leq 0,\\
&g_4(\vec{x}) = 1 - \frac{x_1 + x_2}{1.5} \leq 0\\
&\textit{Ranges of variables}\\
&0.05 \leq d \leq 2.00, \quad 0.25 \leq D \leq 1.30, \quad 2.00 \leq N \leq 15.00
\end{aligned}$$

Instead of choosing strict death penalty which often suffers stagnation, this problem has been formulated by a modified version of barrier penalty approach [58], which works by adding weighted penalty values computed from the deviations from all the constraints to the objective function value. The initial population has been generated

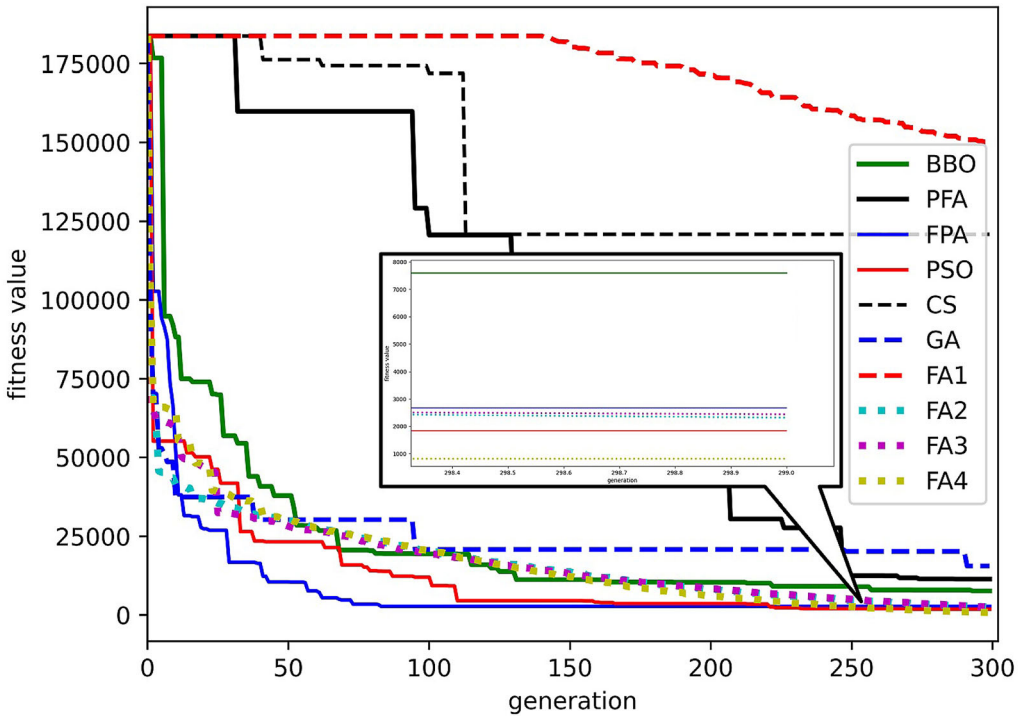

Figure 29. Performance plot of Sum Squares Function for 16-D.

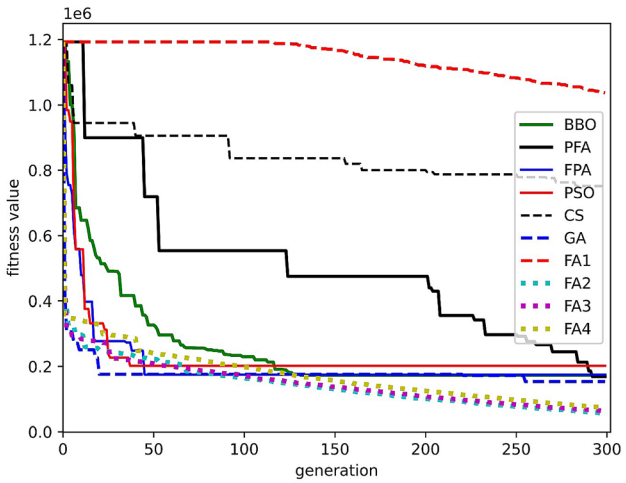

Figure 30. Performance plot of Sum Squares Function for 32-D.

considering the feasible solution space defined by only the individual bounds of the decision variables. The final solution has also been checked for feasibility. The candidate which lies in the last generation of solutions matching all the constraints with the best objective function value has been selected as the solution.

3.1b *Design optimization of 3-bar truss structure*: This problem can be regarded as one of the most studied cases in constrained optimization works for bench-marking. Figure 3 illustrates the shape of the formulated 3-bar truss and the related forces acting on this structure. Here, the design space constitutes of two parameters: the area of bars 1 and 3, which is A_1 and the area of bar 2, which is A_2. The objective of this problem is to minimize the total weight of the structure. While solving this, the optimal design has to satisfy several constraints including stress, deflection, and buckling.This problem can be formulated mathematically as follows:

$$\begin{aligned}
&Consider\ \overrightarrow{x} = [x_1 x_2] = [A_1 A_2],\\
&Minimize f(\overrightarrow{x}) = (2\sqrt{2}x_1 + x_2) \times L,\\
&Subject\quad to\\
&g_1(\overrightarrow{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}{x_1}^2 + 2x_1 x_2} P - \sigma \quad \leq 0,\\
&g_2(\overrightarrow{x}) = \frac{x_2}{\sqrt{2}{x_1}^2 + 2x_1 x_2} P - \sigma \quad \leq 0,\\
&g_3(\overrightarrow{x}) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \quad \leq 0\\
&Ranges\quad of\quad variables\\
&0.00 \leq A_1 \leq 1.00, \quad 0.00 \leq A_2 \leq 1.00\\
&where\quad L = 100\,cm, \quad P = 2N/cm^2, \quad \sigma = 2N/cm^2
\end{aligned}$$

This problem has been similarly modelled by adding penalty values to the specific objective function value. The initial population has been generated considering the feasible solution space defined by both the individual bounds of the decision variables and all the constraints to be

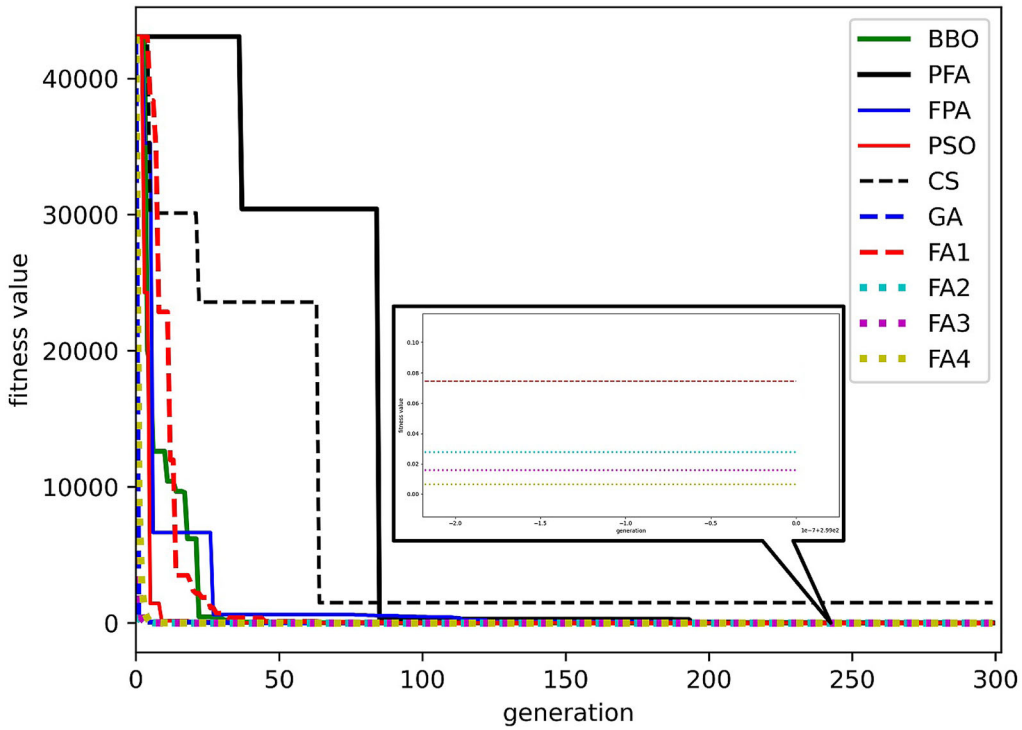

Figure 31. Performance plot of Xin-She-Yang (1) Function for 16-D.

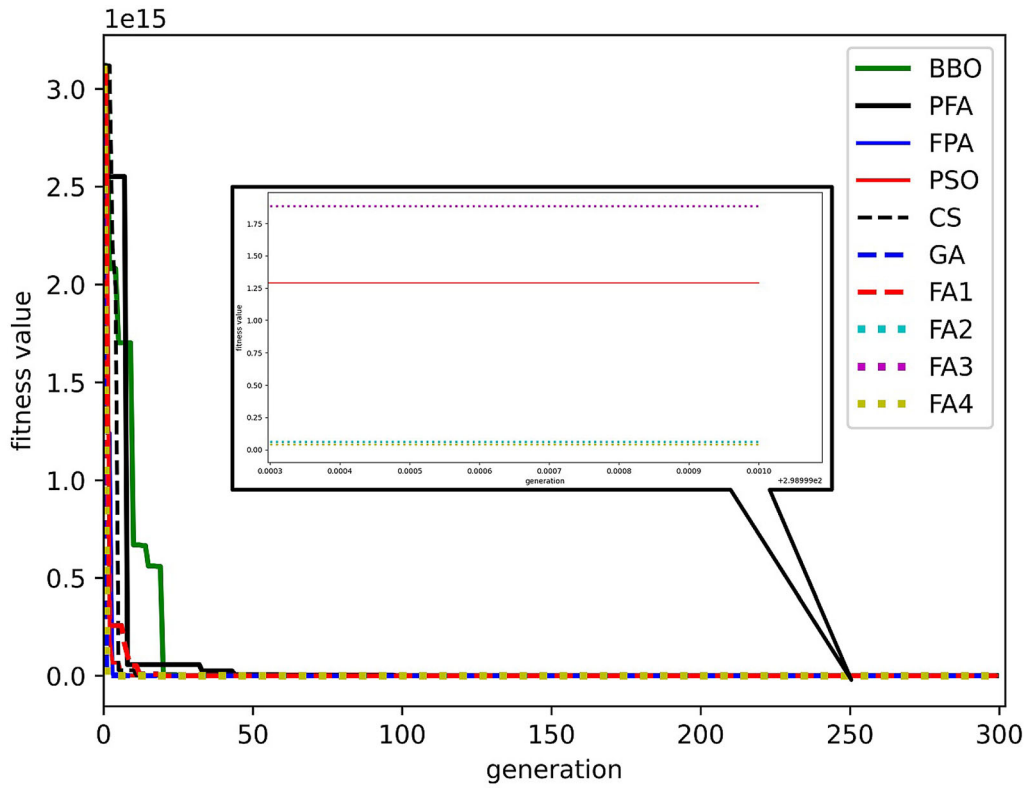

Figure 32. Performance plot of Xin-She-Yang (1) Function for 32-D.

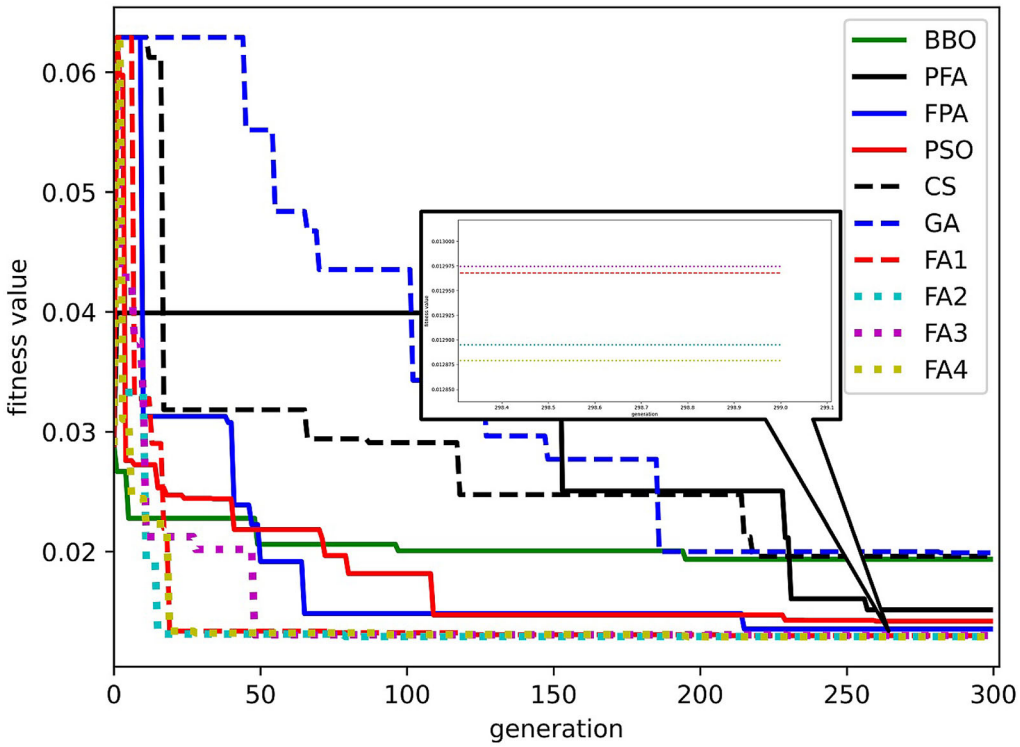

Figure 33. Performance plot of tension-compression spring problem.

Table 6. Comparison results of tension-compression spring problem.

Algorithm	*d*	*D*	*N*	*Optimal weight*
BBO	0.0606	0.4437	11.5313	0.0220628
FPA	0.0524	0.3706	10.7119	0.0129143
PFA	0.0527	0.3662	11.3203	0.0135283
CS	0.0524	0.3474	14.0883	0.0153417
PSO	0.0597	0.5728	5.0634	0.0144132
GA	0.0624	0.6401	5.2427	0.0180378
FA1	0.0500	0.3167	14.2819	0.0128905
FA2	0.0519	0.3570	11.1812	0.0126753
FA3	**0.0518**	**0.3589**	**11.1580**	**0.0126713**
FA4	**0.0518**	**0.3593**	**11.1387**	**0.0126619**

Table 7. Comparison results of 3-bar truss structure problem.

Algorithm	A_1	A_2	*Optimal weight*
BBO	0.7971	0.3875	264.193285
FPA	0.7963	0.3976	264.977011
PFA	0.7980	0.4082	266.533187
CS	0.7980	0.3834	264.046567
PSO	0.7938	0.4016	264.589298
GA	**0.7976**	**0.3903**	**264.618029**
FA1	0.7917	0.4073	264.638401
FA2	0.7971	0.3924	264.694955
FA3	0.7971	0.3914	264.638401
FA4	**0.7886**	**0.4082**	**263.886308**

satisfied. The final solution has been checked for feasibility defined by the bounds and constraints.

2.4 *Statistical tests*

As per the recommendation by Derrac *et al* [59], the non-parametric Wilcoxon rank-sum statistical test with 95% degree of confidence (5% degree of significance) is performed along with experimental evaluations to detect any significant differences between the attained results of different algorithms.

TOPSIS is one of the most popular MCDM tool and has been successfully utilized for decision making across many engineering domains. There are grossly two criteria for relative comparison of all the meta-heuristic algorithms - mean CPU time and mean objective function value computed from 30 independent repeated runs with a specific algorithm. The lower the time and value, the better is the algorithm. The respective mean values for all the

dimensions (2-D, 16-D and 32-D) including the engineering problems are used for comparative study.

3. Results and discussions

It should be mentioned that other than the improvement areas mentioned earlier all the remaining equivalent design and implementation is kept unaltered for all the algorithms in test. The user defined constant parameters are also same for all the algorithms. All the respective parameters and their values are shown in table 2. The results for benchmark objective functions and real engineering problems are discussed in the following subsections.

3.1 *Results for benchmark objective functions*

The population size and maximum number of iteration have been kept 50 and 100 respectively, for all the meta-heuristic algorithms for this lower dimension tests. It is clearly visible from comparison results of tests on 2D objective functions tabulated in table 3, the new PUBG-FA performs better in comparison with the standard FA and GA and it outperforms most of the state-of-the-art algorithms tested in this study. In case of Beale and Schaffer(6), FA3 has been found to be most accurate optimizer, while for Zettl function FA2 has been the topmost performer. For rest of the 2D functions, FA4 with the maximum (70%) weightage/dominance of FA has been proved to be the most suitable blender. It performs satisfactorily well for all the multimodal problems and specifically for functions like Carrom Table and Easom with nearly flat like surfaces containing very small central nadir area. In case of Acley (1) and Zakharov, PFA produces the least objective function value with the new algorithm closely following it. Figures 5 to 14 show the convergence plots of the algorithms for the 2-D objective functions. The convergence

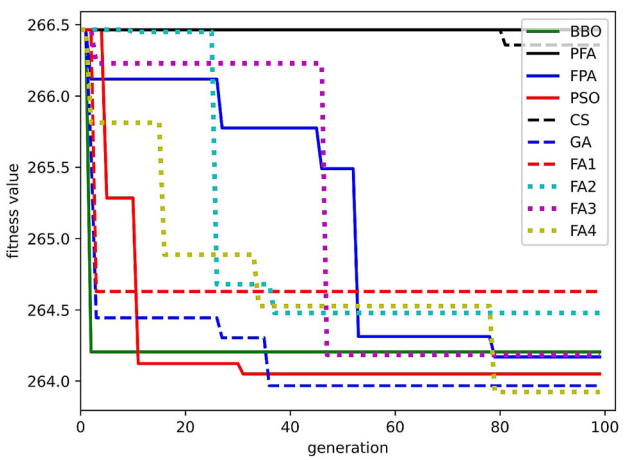

Figure 34. Performance plot of 3-bar truss structure problem.

plots of the Carrom Table function and Easom function reveal that standard GA and standard FA get trapped at local minima, respectively. This clearly raises concerns about using any one specific standard algorithm for the two optimization problems. However, the new algorithm performs well and finds the global minima, converging pretty faster for both the optimization problems. It is clear from the plots that all the algorithms start with the initial same fitness dictated by the same initial population set design. Working with the higher dimensional 16-D benchmark function tests, the new algorithm is found to outperform all the algorithms except for Rastrigin function (table 4). Testing with the same group of functions on 32-D, has produced similar kind of results in favour of the new algorithm except for Rastrigin and Step (2) functions (table 5). The minimum objective values (global minimums) for all the functions tested are 0. For complexly scattered multi-modal functions like Rastrigin and Rosenbrock functions, where all the standard algorithms find it hard to converge to the global minima, the new PUBG-FA shows encouraging results. Figures 15 to 33 show the convergence plots for the 16-D and 32-D objective functions. Competitiveness of different algorithms for Alpine(1) and Rastrigin functions is studied from the respective graphs. They are complex, uniformly distributed multimodal functions with a single global minimum at 0. The standard optimization algorithms are trapped at local minima whereas the proposed FA variants avoid trapping into local minima and converge rapidly. The impressive result is due to the seamless hybridization of the two standard algorithms enhancing both exploration and exploitation. The same happens with Step (2) and Sum Squares function, where the PUBG-FA is able to find competitive results while most of the other algorithms including the standard FA and GA seem to suffer from local optimization traps. In case of Rastrigin function alone when tested on 16-D, the new algorithm is preceded by FPA, PSO and BBO in the ascending order of superiority. In the further higher dimension of Rastrigin and Step (2) functions, BBO and FPA has been found the best performer respectively. In all the functions of higher dimensions too, PUBG-FA has outperformed the standard algorithms FA and GA quite consistently. The fast convergence rate of the new algorithm is very much evident from all the convergence plots. The success of PUBG-FA is due to the continuous exchange of useful information between two complete sets of different algorithms.

Even though, the variant with highest w (70%) as represented by FA4 is the overall best performer among all, FA2 with the least weightage of FA (30%) shows commendable performance when considered in terms of both accuracy and speed of convergence. FA2 has seen comparatively higher success rates in the higher dimensions. The most remarkable result is achieved while optimizing the 16-D and 32-D Sphere, Step (2) and Sum Squares functions, where it could draw a clear demarcation line of

Table 8. *p*-values obtained in Wilcoxon rank-sum tests on 2-D functions with 5% significance.

	BBO	FPA	PFA	CS	PSO	GA	FA1	FA2	FA3	FA4
Ackley (1)	1.68E-04	1.66E-04	1.69E-04	1.68E-02	3.68E-04	1.76E-04	1.09E-04	2.12E-04	1.09E-04	1.13E-04
Beale	1.66E-03	1.68E-04	1.28E-04	2.23E-02	4.09E-04	1.76E-04	1.87E-04	6.65E-04	1.87E-04	4.39E-04
Carrorm Table	1.76E-03	1.21E-03	1.23E-03	2.23E-02	1.21E-03	1.18E-03	5.10E-03	6.62E-04	5.10E-03	4.43E-03
Easom	1.56E-03	1.18E-03	1.38E-03	2.46E-02	1.18E-03	7.44E-03	8.44E-03	1.84E-02	2.35E-02	7.06E-03
Goldstein-Price	1.57E-02	4.45E-04	1.31E-03	1.98E-02	4.45E-04	7.44E-03	1.94E-03	1.94E-03	1.78E-03	1.18E-03
Michalewicz	1.20E-02	1.32E-03	3.22E-04	1.76E-02	3.22E-04	5.54E-04	3.40E-04	3.40E-04	6.75E-04	6.54E-04
Schaffer (6)	1.20E-02	7.16E-03	1.62E-04	1.99E-02	1.62E-04	5.54E-04	4.24E-04	4.24E-04	1.83E-04	4.24E-04
Xin-She-Yang (3)	3.12E-03	5.17E-04	1.69E-05	1.47E-04	1.69E-05	1.45E-05	2.09E-04	2.09E-04	9.42E-05	2.09E-04
Zettl	1.31E-02	9.18E-04	1.77E-04	1.65E-02	1.77E-04	6.52E-04	5.43E-04	3.97E-04	5.43E-04	1.20E-04
Zakharov	1.12E-03	1.57E-04	1.57E-04	1.51E-02	1.57E-04	2.57E-04	5.65E-03	8.11E-03	4.10E-03	7.72E-03

Table 9. p-values obtained in Wilcoxon rank-sum tests on 16-D functions with 5% significance.

	BBO	FPA	PFA	CS	PSO	GA	FA1	FA2	FA3	FA4
Alpine (1)	1.00E-03	3.36E-02	2.12E-03	4.16E-03	2.07E-03	5.66E-03	2.54E-02	1.33E-02	3.37E-02	3.05E-02
Dixon-Price	3.44E-03	3.53E-03	2.07E-02	3.44E-02	3.32E-02	2.44E-02	2.79E-02	1.90E-02	1.61E-03	1.17E-02
Levy	1.67E-03	4.11E-03	9.80E-03	2.40E-02	1.59E-02	6.32E-03	6.58E-03	1.33E-02	1.44E-02	4.37E-03
Rastrigin	2.92E-02	1.86E-02	2.54E-02	2.45E-02	8.91E-03	2.21E-02	2.68E-02	1.65E-03	8.12E-03	1.91E-02
Rosenbrock	1.22E-02	1.15E-02	2.90E-02	2.55E-02	3.09E-02	1.79E-03	2.45E-02	1.36E-02	1.17E-02	2.27E-02
Sphere	6.00E-03	3.03E-02	1.78E-03	3.11E-02	2.64E-02	3.10E-02	3.26E-02	2.74E-02	4.53E-03	8.83E-03
Step (2)	1.26E-02	9.23E-03	3.00E-03	7.41E-03	1.88E-02	1.91E-02	2.51E-03	2.11E-02	1.42E-02	1.67E-02
Sum Squares	8.98E-03	3.37E-02	2.71E-02	1.43E-02	2.79E-03	3.43E-02	1.76E-02	1.40E-02	1.60E-02	1.31E-02
Xin-She-Yang (1)	2.10E-02	2.55E-02	1.95E-02	8.49E-03	2.54E-02	8.36E-03	5.53E-04	2.75E-02	5.19E-03	1.88E-02

Table 10. p-values obtained in Wilcoxon rank-sum tests on 32-D functions with 5% significance.

	BBO	FPA	PFA	CS	PSO	GA	FA1	FA2	FA3	FA4
Alpine (1)	3.43E-03	7.65E-03	1.74E-02	3.43E-03	2.08E-03	5.43E-03	1.43E-03	2.91E-03	9.88E-03	7.75E-03
Dixon-Price	1.89E-03	1.76E-03	5.11E-03	2.33E-03	4.30E-03	1.38E-03	7.56E-03	1.78E-03	7.89E-03	6.89E-03
Levy	1.53E-03	1.19E-03	2.90E-03	5.67E-02	1.21E-03	4.34E-03	5.44E-03	7.87E-03	4.17E-03	6.34E-03
Rastrigin	1.15E-02	1.81E-02	3.18E-02	1.34E-02	1.90E-02	2.74E-02	4.55E-02	2.12E-02	2.32E-02	2.44E-02
Rosenbrock	5.60E-02	6.56E-04	3.31E-02	1.77E-02	1.28E-03	7.72E-03	1.43E-03	1.47E-03	1.65E-03	1.00E-03
Sphere	1.17E-02	1.07E-03	3.97E-03	1.79E-02	6.62E-03	7.54E-03	2.21E-03	5.64E-03	6.12E-03	6.55E-03
Step (2)	1.65E-03	1.99E-03	3.97E-03	1.67E-02	1.00E-03	6.58E-03	2.19E-03	3.76E-03	5.61E-03	2.24E-03
Sum Squares	3.09E-03	1.16E-03	3.54E-03	1.74E-02	5.58E-03	7.46E-03	2.53E-03	5.34E-03	7.71E-03	1.09E-03
Xin-She-Yang (1)	1.52E-03	1.85E-03	2.44E-03	1.89E-02	7.72E-03	8.78E-03	3.22E-03	9.12E-03	5.88E-03	1.20E-03

quality between itself and the two standard algorithms used as the constituents - FA and GA.

The average time taken and standard deviations are also far less than that of the standard FA and for lesser w variant it is very close to GA. The number of functional evaluations for all the algorithms is logged and found same for all the algorithms and it matches with the theoretically obtainable values ($D \times N$) i.e. 5000 and 60000 for the 2-D and higher dimensional functions (16-D and 32-D) respectively.

The time complexity of the new algorithm is studied by analytical method alongside plotting with a close watch on the reported time of execution in logs. Considering N as the population size the time complexity for the standard GA and FA are in the order of $O(\frac{9N}{2} + 2N\log(2N))$ and $O(\frac{N}{2} \times (N-1) + N\log(N))$ respectively. The newly developed PUBG-FA maintains an intermediate order of time as regards the standard algorithms, which can be computed as $O(\frac{9N}{4} + \frac{N}{4} \times (\frac{N}{2} - 1) + 2N\log(2N))$.

Table 11. Results of TOPSIS conducted on all the algorithms. Equal weights for all functions. Weights for time:objective value = 1:3.

	2-D benchmarking		16-D benchmarking		32-D benchmarking		Engineering optimization		**Overall performance**	
Algorithm	Score	Rank	**Score**	**Rank**	**Score**	**Rank**	**Score**	**Rank**	**Score**	**Rank**
BBO	3.33E-01	10	9.18E-01	2	9.59E-01	1	4.76E-01	9	4.95E-01	10
FPA	8.19E-01	6	9.14E-01	3	8.87E-01	4	4.46E-01	10	8.34E-01	6
PFA	6.22E-01	9	6.13E-01	8	5.84E-01	8	8.93E-01	2	6.15E-01	8
CS	6.85E-01	8	3.29E-01	10	3.21E-01	10	5.65E-01	6	5.38E-01	9
PSO	8.94E-01	5	9.46E-01	1	9.18E-01	2	8.47E-01	4	9.05E-01	**3**
GA	9.56E-01	2	8.00E-01	7	7.98E-01	7	5.60E-01	7	8.68E-01	4
FA1	6.98E-01	7	5.20E-01	9	5.16E-01	9	5.56E-01	8	6.24E-01	7
FA2	9.69E-01	1	8.97E-01	4	9.09E-01	3	9.38E-01	1	9.35E-01	**1**
FA3	9.47E-01	3	8.67E-01	5	8.69E-01	5	8.48E-01	3	9.08E-01	**2**
FA4	9.10E-01	4	8.11E-01	6	8.10E-01	6	7.46E-01	5	8.62E-01	5

Figure 4 shows the plotting of time complexity of the new algorithm (FA3) along with that of the constituents used. It is evident from the mean CPU time tabulated in the table 4, all the variants of the new algorithm has an intermediate time complexity between standard GA and FA1 (standard FA). The fitness convergence plots against elapsed computation time for all the function tests and case studies are separately available as supplementary files (Figures. S1 to S30).

3.2 *Results of engineering case study problems*

The new algorithm has shown clear supremacy in dealing with constrained optimization problems when entrusted to solve two real engineering case studies. Population size and maximum number of iterations have been set as 200 and 300 respectively for the tension-compression spring problem having three variables. The FA4 variant of PUBG-FA has been able to expose the best solution with optimal weight satisfying all the constraints. The comparison results obtained in this study are tabulated in table 6. Similarly, the 3-bar truss optimization problem has been solved by all the algorithms using population size and maximum iteration number as 50 and 100 respectively. Here also, FA4 has been the topmost performer producing the least objective value. The results shown in table 7 indicate a total supremacy of all the variants of the new algorithm in the context of this particular case study. The performance plots for the engineering case studies are shown in figures 33 and 34. This proves that the new algorithm is aptly suitable in handling real world constraints besides achieving global optimization.

3.3 *Results of statistical tests*

The results of non-parametric Wilcoxon rank-sum tests are tabulated in tables 8, 9, and 10 for 2-D, 16-D and 32-D benchmark functions respectively. The p-values are all below 0.05 (with 5% significance) and thus confirm that there is no significance differences in the results obtained in independent trials. The final ranks obtained by applying TOPSIS considering all the test scenarios are shown in table 11. It shows that FA2 and FA3 variants of PUBG-FA are the two top performing algorithms followed by PSO and GA. The other variant of the new algorithm - FA4 is assigned the 5th overall rank. Equal weights are given for all the objective functions and problems whereas ratio of weights of computation time to objective value is chosen as 1:3.

4. Conclusions

In this paper, a new algorithm PUBG-FA is proposed, which first partitions the population in two compartments for running a slightly modified FA and standard GA. Then it unifies the worked-upon sub-populations of two compartments and finally ranks the fireflies before going into next generation. The new algorithm is experimented with three variants of weightage, w which determines the percentage of total population belonging to FA chamber. This is tested on 19 different benchmark optimization functions, the first 10 of which are used as 2-D functions and the rest are run separately as 16-D and 32-D functions. The results, when compared with the standard FA and GA, show the algorithm is significantly more efficient with faster convergence than the standard FA and lower minimal values than the two. The new algorithm is also compared with some of the state-of-the-art meta-heuristic algorithms on the same functions and encouraging results are observed. A popular MCDM tool TOPSIS reveals that the new variants are very competitive and successfully outperform some renowned algorithms in majority of the cases.

Meta-heuristic algorithms are not panacea. It has been exhibited that different optimization problems suit different techniques ranging from simple classical optimization methods to complex meta-heuristic and population-based

evolutionary algorithms. The methodology adopted in this study can be seen as a new approach of dealing with multiple population based meta-heuristic algorithms collaboratively for solving a particular optimization problem. This work has some prompting directions pertaining to the future works as follows:

- The same hybridization schema can be rigorously explored in future considering other combination of some of the top performing evolutionary algorithms like PSO, FPA, BBO, etc.
- The new PUBG-FA can be further tailored for new practical problems associated with social and engineering domain, e.g., preservation of privacy in social networks, the multicast vehicle routing, and parameter tuning in training of machine learning and artificial intelligence models.

Abbreviations

GA	Genetic algorithm
FA	Firefly algorithm
PUBG-FA	Partition cum unification based genetic - FA
PSO	Particle swarm optimization
CS	Cuckoo search
FPA	Flower pollination algorithm
PFA	Pathfinder algorithm
BBO	Bio-geography based optimization
MCDM	Multi-criteria decision making
TOPSIS	Technique for order of preference by similarity to ideal solution

References

[1] Xin-She Yang. *Nature-inspired metaheuristic algorithms*. Luniver press, 2010

[2] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information sciences*, 237:82, 2013

[3] John H Holland. Genetic algorithms and adaptation. In *Adaptive Control of Ill-Defined Systems*, pp 317. Springer, 1984

[4] Kalyanmoy Deb. An introduction to genetic algorithms. *Sādhanā*, 24(4-5):293, 1999

[5] Tansel Dokeroglu, Ender Sevinc, Tayfun Kucukyilmaz, and Ahmet Cosar. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, 137:106040, 2019

[6] Kashif Hussain, Mohd Najib Mohd Salleh, Shi Cheng, and Yuhui Shi. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52(4):2191, 2019

[7] Xin-She Yang. Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms*, p 169. Springer, 2009

[8] Waqar A Khan, Nawaf N Hamadneh, Surafel L Tilahun, and Ngnotchouye J M. A review and comparative study of firefly algorithm and its modified versions. *Optimization Algorithms-Methods and Applications*, p 281, 2016

[9] Iztok Fister, Iztok Fister Jr, Xin-She Yang, and Janez Brest 2013. A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*, 13:34, 2013

[10] Nilanjan Dey. *Applications of firefly algorithm and its variants*. Springer, Berlin, 2020

[11] Mahmood Reza Shakarami and Reza Sedaghati. A new approach for network reconfiguration problem in order to deviation bus voltage minimization with regard to probabilistic load model and dgs. *Int. J. Electr. Comput. Energ. Electr. Commun. Eng*., 8(2):430, 2014

[12] Abdollah Kavousi-Fard, Haidar Samet, and Fatemeh Marzbani. A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting. *Expert systems with applications*, 41(13):6047, 2014

[13] Xiaoyu Lin, Yiwen Zhong, and Hui Zhang. An enhanced firefly algorithm for function optimisation problems. *International Journal of Modelling, Identification and Control*, 18(2):166, 2013

[14] Amir Hossein Gandomi, X-S Yang, S Talatahari, and Amir Hossein Alavi. Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation*, 18(1):89, 2013

[15] Mohd Herwan Sulaiman, Hamdan Daniyal, and Mohd Wazir Mustafa. Modified firefly algorithm in solving economic dispatch problems with practical constraints. In *2012 IEEE International Conference on Power and Energy (PECon)*, pp 157. IEEE, 2012

[16] Bin Wang, Dong-Xu Li, Jian-Ping Jiang, and Yi-Huan Liao. A modified firefly algorithm based on light intensity difference. *Journal of Combinatorial Optimization*, 31(3):1045, 2016

[17] Shuhao Yu, Shoubao Su, Qingping Lu, and Li Huang. A novel wise step strategy for firefly algorithm. *International Journal of Computer Mathematics*, 91(12):2507, 2014

[18] Amit Kumar Ball, Shibendu Shekhar Roy, Dakshina Ranjan Kisku, Naresh Chandra Murmu, and Leandro dos Santos Coelho. Optimization of drop ejection frequency in ehd inkjet printing system using an improved firefly algorithm. *Applied Soft Computing*, 94:106438, 2020

[19] Abhishek Ghosh Roy, Pratyusha Rakshit, Amit Konar, Samar Bhattacharya, Eunjin Kim, and Atulya K Nagar. Adaptive firefly algorithm for nonholonomic motion planning of car-like system. In *2013 IEEE Congress on Evolutionary Computation*, pp 2162. IEEE, 2013

[20] Shuhao Yu, Shenglong Zhu, Yan Ma, and Demei Mao. Enhancing firefly algorithm using generalized opposition-based learning. *Computing*, 97(7):741, 2015

[21] MJ Kazemzadeh-Parsi. A modified firefly algorithm for engineering design optimization problems. *Iranian Journal of Science and Technology. Transactions of Mechanical Engineering*, 38(M2):403, 2014

[22] Mohammad Javad Kazemzadeh-Parsi, Farhang Daneshmand, Mohammad Amin Ahmadfard, and Jan Adamowski. Optimal remediation design of unconfined contaminated aquifers based on the finite element method and a modified

firefly algorithm. *Water Resources Management*, 29(8):2895, 2015

[23] Sirus Mohammadi, Babak Mozafari, Soodabeh Solimani, and Taher Niknam. An adaptive modified firefly optimisation algorithm based on hong's point estimate method to optimal operation management in a microgrid with consideration of uncertainties. *Energy*, 51:339, 2013

[24] Tahereh Hassanzadeh and Hamidreza Rashidy Kanan. Fuzzy fa: a modified firefly algorithm. *Applied Artificial Intelligence*, 28(1):47, 2014

[25] Sankalap Arora, Sarbjeet Singh, Satvir Singh, and Bhanu Sharma. Mutated firefly algorithm. In *2014 International Conference on Parallel, Distributed and Grid Computing*, p 33. IEEE, 2014

[26] Sankalap Arora and Satvir Singh. Performance research on firefly optimization algorithm with mutation. In *International conference, computing & systems*, 2014

[27] Wen-chuan Wang, Lei Xu, Kwok-wing Chau, and Dong-mei Xu. Yin-yang firefly algorithm based on dimensionally cauchy mutation. *Expert Systems with Applications*, 150:113216, 2020

[28] Hu Peng, Wenhua Zhu, Changshou Deng, and Zhijian Wu. Enhancing firefly algorithm with courtship learning. *Information Sciences*, 543:18, 2021

[29] Nan Tong, Qiang Fu, Caiming Zhong, and Pengjun Wang. A multi-group firefly algorithm for numerical optimization. In *Journal of Physics: Conference Series*, vol 887, p 012060. IOP Publishing, 2017

[30] Lingyun Zhou, Lixin Ding, Maode Ma, and Wan Tang. An accurate partially attracted firefly algorithm. *Computing*, 101(5):477, 2019

[31] Adil Baykasoğlu and Fehmi Burcin Ozsoydan. An improved firefly algorithm for solving dynamic multidimensional knapsack problems. *Expert Systems with Applications*, 41(8):3712, 2014

[32] Jingsen Liu, Yinan Mao, Xiaozhen Liu, and Yu Li. A dynamic adaptive firefly algorithm with globally orientation. *Mathematics and Computers in Simulation*, 2020

[33] Iztok Fister, Xin-She Yang, Janez Brest, and Iztok Fister Jr. Modified firefly algorithm using quaternion representation. *Expert Systems with Applications*, 40(18):7220, 2013

[34] Hui Wang, Wenjun Wang, Xinyu Zhou, Hui Sun, Jia Zhao, Xiang Yu, and Zhihua Cui. Firefly algorithm with neighborhood attraction. *Information Sciences*, 382:374, 2017

[35] Sh M Farahani, AA Abshouri, B Nasiri, and MR2011 Meybodi. A gaussian firefly algorithm. *International Journal of Machine Learning and Computing*, 1(5):448, 2011

[36] Xin-She Yang. Firefly algorithm, levy flights and global optimization. In *Research and development in intelligent systems XXVI*, p 209. Springer, 2010

[37] Lyes Tighzert, Cyril Fonlupt, and Boubekeur Mendil. A set of new compact firefly algorithms. *Swarm and Evolutionary Computation*, 40:92, 2018

[38] Jinran Wu, You-Gan Wang, Kevin Burrage, Yu-Chu Tian, Brodie Lawson, and Zhe Ding. An improved firefly algorithm for global continuous optimization problems. *Expert Systems with Applications*, 149:113340, 2020

[39] Jitin Luthra and Saibal K Pal. A hybrid firefly algorithm using genetic operators for the cryptanalysis of a monoalphabetic substitution cipher. In *2011 World congress on information and communication technologies*, p 202. IEEE, 2011

[40] A Rahmani and SA MirHassani. A hybrid firefly-genetic algorithm for the capacitated facility location problem. *Information Sciences*, 283:70, 2014

[41] Shubhendu Kumar Sarangi, Rutuparna Panda, Sabnam Priyadarshini, and Archana Sarangi. A new modified firefly algorithm for function optimization. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, p 2944. IEEE, 2016

[42] İbrahim Berkan Aydilek, İzzettin Hakan Karaçizmeli, Mehmet Emin Tenekeci, Serkan Kaya, and Abdülkadir Gümüşçü. Using chaos enhanced hybrid firefly particle swarm optimization algorithm for solving continuous optimization problems. *Sādhanā*, 46(2):1, 2021

[43] Aref Yelghi and Cemal Köse. A modified firefly algorithm for global minimum optimization. *Applied Soft Computing*, 62:29, 2018

[44] Kadavy Tomas, Pluhacek Michal, Viktorin Adam, and Senkerik Roman. Firefly algorithm enhanced by orthogonal learning. In *Computer Science On-line Conference*, p 477. Springer, 2018

[45] Yu-Pei Huang, Xiang Chen, and Cheng-En Ye. A hybrid maximum power point tracking approach for photovoltaic systems under partial shading conditions using a modified genetic algorithm and the firefly algorithm. *International Journal of Photoenergy*, 2018, 2018

[46] Russell Eberhart and James Kennedy. Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948. Citeseer, 1995

[47] Xin-She Yang and Suash Deb. Cuckoo search via lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)*, p 210. IEEE, 2009

[48] R Indumathy, S Uma Maheswari, and G Subashini. Nature-inspired novel cuckoo search algorithm for genome sequence assembly. *Sādhanā*, 40(1):1, 2015

[49] Xin-She Yang. Flower pollination algorithm for global optimization. In *International conference on unconventional computing and natural computation*, p 240. Springer, 2012

[50] Hamza Yapici and Nurettin Cetinkaya. A new meta-heuristic optimizer: pathfinder algorithm. *Applied soft computing*, 78:545, 2019

[51] Dan Simon. Biogeography-based optimization. *IEEE transactions on evolutionary computation*, 12(6):702, 2008

[52] Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2019

[53] Amir Parnianifard, Ratchatin Chancharoen, Gridsada Phanomchoeng, and Lunchakorn Wuttisittikulkij. A new approach for low-dimensional constrained engineering design optimization using design and analysis of simulation experiments. *International Journal of Computational Intelligence Systems*, 13(1):1663, 2020

[54] Jasbir Singh Arora. *Introduction to optimum design*. Elsevier, 2004

[55] Young-Jou Lai, Ting-Yun Liu, and Ching-Lai Hwang. Topsis for modm. *European journal of operational research*, 76(3):486, 1994

[56] Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimization problems. *arXiv preprint*arXiv:1308.4008, 2013

[57] Sudhanshu K Mishra. Some new test functions for global optimization and performance of repulsive particle swarm method. *Available at SSRN 926132*, 2006

[58] Carlos A Coello Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113, 2000

[59] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3, 2011