THE PARTICLE SWARM MULTI-SEARCH SPACE AUGMENTATION OPTIMIZATION

Derrick Donkor

Abstract

The proposed Particle Swarm Optimization (PSO) variant uses a search space with a non-overlapping distinct search space for each particle in the population in the exploration of the optimum solution. What is normally done for a reduction in swarm size and achieving a much quicker response in PSO is to manually set the swarm size and other auxiliary constants through trial and error. An algorithm is proposed which assigns each particle to a unique non-overlapping finite search space and aggregates all particles position to form the solution at every functional evaluation. This assignment of the particles to a finite distinct search space is suitable for quick convergence with less iteration and less particle size comparatively. The theoretical basis is provided for the proposed algorithm and empirical studies are conducted to compare the proposed algorithm with other selected optimization algorithms on reference benchmark test functions.

Keywords: Particle swarm optimization, search space, multi-search space, augmentation and global solution.

1. Introduction

Ever since its introduction in 1995 by (Kennedy and Eberhart, 1995), PSO has emerged as one of the most popular metaheuristic optimization algorithms. From (Poli et al., 2007) PSO was influenced by Heppner Grenander's work (Heppner et al., 1990) and involved analogues of bird flocks searching for corn and these soon developed into a pioneering optimization method, Particle Swarm Optimization (PSO). The proposed algorithm can be considered a variant of PSO. The real world human society presents numerous problems and challenges on the day to day application of science and technology. Modeling these problems as functional optimization problems is a great way of solving them at a much lower cost economically and computationally. Conventional metaheuristic optimizers are generally incapable of attaining highly accurate results under lesser iterations. The proposed optimization technique presents a more elegant approach in finding optimal solutions with an upgrade and redesign in some aspects of the canonical PSO algorithm.

The first introduction of the Particle Swarm Algorithm came from Kennedy and Eberhart (1995), in that paper, it outlined the relationships between particle swarm optimization, artificial life and genetic algorithms. The algorithm was inspired by natural swarm behaviors such as those exhibited by bird flocks. Another variant of the algorithm also came from Kennedy and Eberhart (1997), which was in the form of Binary Particle Swarm, as the earliest of its kind which operates on binary string velocity and position instead of real numbers. In this PSO variant, the velocity is utilized as a probability threshold to determine whether a given number in the binary string of the current position should be toggled to a zero or a one, if the sigmoid of the given number is greater than or less than a random number respectively. Agrafiotis and Cedeño (2002) proposed a roulette wheel based probabilistic mapping approach for normalizing particle location per its floating-point value. It is a feature selection algorithm for correlating topological activity and property based on the particle swarm property. Zhou et al. (2021) proposed pioneering work based on the diversity evaluation on particle swarms. It was illustrated that fewer particles ensure quick attainability to optimal solution whilst more of the particles improve exploration capacity. The diversity is attained by hash table technique and novel encoding of subspaces of search space. Zhu et al. (2022) proposed a dynamic multi-search PSO variant which is composed of particles divided into sub-swarms with a center-learning update strategy. The center-learning strategy is such that all the other particles will learn from the optimal particle in their swarm; an alternative learning

factor is given to determine the particle learning strategy. In this research it can be deduced that there is a high certainty in obtaining the optimal solution.
The proposed PSO variant provides a pioneering way to estimate the optimal solution such that the value for the optimal solution is the combination of the individual location of each particle from the swarm. One major advantage is the high precision and quick convergence to the optimal solution of convex functions.

The contributions of the proposed paper are as follows;

1. To present an optimization algorithm which performs functional optimization under lesser iterations, a much smaller particle population and results in a better optimal solution compared to other benchmark functions presented in Section 3.2.

2. The proposed algorithm involves particles getting assigned the decimal place values of the initial solution.

The paper is organized as follows; Section 2 gives the review of the preliminary concepts, section 3 contains the related works, the proposed algorithm is presented in Section 4, numerical experiment is performed in Section 5 and conclusions in Section 6.

2. Preliminary Concepts

There has been a long list of evolutionary algorithms based on PSO. This paper seeks to model each particle's location as a part of the optimal solution. Majority of the research done focuses more on inertia constant setting, acceleration constant setting, velocity initialization, position initialization and update rule modification, particle swarm topology setting, PSO hybridization and PSO composition. This paper goes a step further to redesign a new mechanism for particle association. In the subsequent subsections, relevant concepts are explained and the general concept of PSO outlined. Also, a brief overview of the proposed algorithm is presented.

2.1. Overview of PSO

The PSO technique as a typical heuristics algorithm needs direct manipulation of at least one of its elementary features: the population size, position, acceleration and the topology of the particles.
PSO uses a number of particles which are placed in some problem search space, with an objective function evaluated for an optimal solution. The particles are updated with their historic performance with little random perturbations and then set in a given rate of motion which depends on whether or not it is close enough to an optimal solution. The position update takes place after all particles have been checked for an optimal solution. The three parameters of each particle in the swarm, given by the current position, previous best position and the velocity are updated in each step. The current position is basically a point in the problem space and if it is evaluated to be more optimal than any position attained by a particle so far on the objective function then it is set as the best position, replacing the previous best position. Also, the velocity component is considered as the step distance to be moved to by the particle.

2.2. Overview of PSMSAO

The proposed algorithm is a variant of PSO but unlike PSO where all the particles compete to find an optimum solution; this algorithm combines the particles in a more harmonious way, such that each particle forms part of the optimal solution. The fundamental difference between PSMSAO and PSO is how particles coordinate to find the optimal solution. The problem search space of PSO is made up of particles are all in one swarm space. On the other hand, PSMSAO has each particles confined to a unique swarm space, where each particle's optimal location in respective swarms corresponds to a particular decimal place value of the overall optimal solution. For instance; if an optimal solution is found to be $a.bc$, such that a, b and c are strictly single digit integer values, eg. 2.45, then with three particles, the first would have a value of a, the second a value of b and the third a value of c. From the example above there are 3 particles with 1-dimensional swarm space, and each particle generally contributes to the formation of the optimal solution.

3. The proposed Algorithm (PSMSAO)

The optimizer proposed has its performance evaluated by functional optimization approaches, specifically with benchmark test functions from CEC2022. Detailed algorithmic and block diagram description is presented in Sections 3.4.1 and 3.4.2 respectively. Some benchmark functions were used for the comparison between this proposed concept and other optimization techniques. Graphs on the metric measurement are presented to illustrate its performance. Multiple tests are conducted and the results given in Section 5, on the behavior of PSMSAO and other techniques under a couple of dimensions.

3.1. Mathematical Formulation of the Proposed Algorithm

Given that $v_i{}^m$ is the velocity of particle i at iteration m.
φ_1, φ_2 are the acceleration constants.
ε is the inertia constant, δ_m is the random number at iteration m.
$p, g \wedge x$ are local best, global best and position respectively.
The velocity update rule and the position update rule is given respectively by (1) and (2).

$$v_i{}^m = \varepsilon v_i{}^{m-1} + \varphi_1 \delta_m (p - x) + \varphi_2 \delta_m (g - x) \quad (1)$$

$$x_i{}^m = x_i{}^{m-1} + v_i{}^m \quad (2)$$

The Particles' position is constrained to a unique decimal place value in the attained optimal value. At each iteration, each particles' position changes to affect its own part of the optimal value which is the decimal place value of the overall optimal value. PSO has a single solution set which is a finite continuous set unlike PSMSAO whose solution set is theoretically a finite discrete, non-overlapping set. Considering one dimensional root estimation on three arbitrary particles, the particles can have a search space given by;

$S_1 =$ **{-Z,…, -2,-1,0, 1, 2,3,4,5,6,7,8,9, …, Z}** (3)

For a considerably large number **Z**, another

$S_2 =$ **{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}** (4)

and the third particle

$S_3 =$ **{0.00, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09}** (5)

The first particle would be a set of integers and the ith particle apart from the first would inductively have

$$S_i = \{\mathbf{0} \times \tfrac{1}{10^{i-1}}, \mathbf{1} \times \tfrac{1}{10^{i-1}}, \mathbf{2} \times \tfrac{1}{10^{i-1}}, \mathbf{3} \times \tfrac{1}{10^{i-1}}, \mathbf{4} \times \tfrac{1}{10^{i-1}}, \mathbf{5} \times \tfrac{1}{10^{i-1}}, \mathbf{6} \times \tfrac{1}{10^{i-1}}, \mathbf{7} \times \tfrac{1}{10^{i-1}}, \mathbf{8} \times \tfrac{1}{10^{i-1}}, \mathbf{9} \times \tfrac{1}{10^{i-1}}\} \quad (6)$$

where $i \leq N$ for root estimation on N particles. Practically increasing N reduces the approximation error, hence N should be as small as possible but not to cause an approximation error of more than or equal to 0.05. Since each particle is dedicated to one and only one decimal place value, the periodic aggregation which is basically the augmentation process of the particles positions in verifying the optimal solution at each functional evaluation ensures information sharing between the particles. Generalizing the one-dimensional roots estimation on three particles to N-dimensions on M-particles. The set of particles given by S is defined as;

$$S = \{S_1, S_2, S_3, \dots, S_m\} \quad (7)$$

Where

$$S_1 = \{S_{11}, S_{12}, S_{13}, \dots, S_{1n}\} \quad (8)$$

$$S_2 = \{S_{21}, S_{22}, S_{23}, \dots, S_{2n}\} \quad (9)$$

$$\vdots$$

$$S_m = \{S_{m1}, S_{m2}, S_{m3}, \dots, S_{mn}\} \quad (10)$$

The nature of each element can be obtained from 3; The composition of evaluated root 'r' is as followed below. Given that Z is the central value between the maximum and minimum value of S_1

$$a \in \{2, 3, 4, .., m\} \quad (11)$$

$$k \in \{1,2,3, \dots, 2Z\} \quad (12)$$

$$j \in \{1,2,3,4, \dots, 10\} \quad (13)$$

'j' can be an unordered non-consecutive iterable element under the summation operation, where the value of j can be repeated. Which means the iteration under summation can start from a random position in the set go through other random positions then finally end up at a random position on the set, instead of strictly following the the positions in succession. Then roots 'r' is given by;

$$r = S_{1k} + \sum_{a=2,j}^{m} S_{aj} \quad (14)$$

S_{aj} means jth value in ath solution space

S_{1k} means kth value in first solution space

$$X_a = S_{aj} \quad (15)$$

$$X_1 = S_{1k} \quad (16)$$

$$r = S_{1k} + \sum_{a=2}^{m} X_a \quad (17)$$

$$r = X_1 + \sum_{a=2}^{m} X_a \quad (18)$$

$$X_1 \in R \quad (19)$$

$$i \in \{0,1,2,3, \dots, 9\} \quad (20)$$

'i' has similar property as 'j'.

$$\sum_{a=2}^{m} X_a = \sum_{a=2}^{m} \frac{i}{10^{a-1}} \quad (21)$$

Where X is the augmented position of optimal solution on the solution space S

4.2. Conceptual View of PSO and PSMSAO

4.2.1 Simple Representation of PSMSAO optimal solution Discovery Mechanism

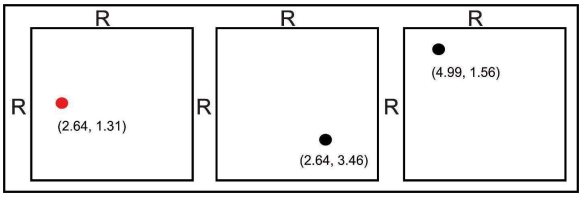

Figure 1(a). The representation of PSO optimal solution search activity

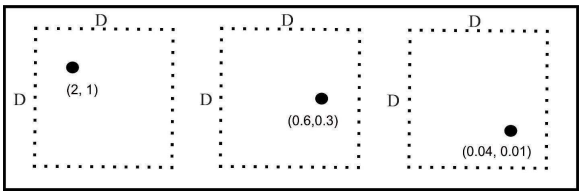

Figure 1(b). The representation of PSMSAO optimal solution search activity

Figure 1(a) is a swarm space, which is a coordinate plane with a real axis $R \times R$ on both axis likewise Figure 1(b) is a swarm space and also a coordinate plane with a discrete axis $D \times D$ on both axis.
The optimal solution is (2.63, 1.31) in both Figure 1(a) and 1(b); there are three particles in both figure 1(a) and 1(b). For each iteration, a single resource is utilized in estimating the optimal solution for PSO but for the proposed algorithm, all the resources are synergized, by maintaining one particle as the whole number part of the overall roots and fitting the remaining particles as the remaining successive decimal place values for the overall roots, in estimating the optimal solution. For instance, as the optimal solution is (2.63, 1.31), the first particle takes (2, 1), the second particle takes (0.6, 0.3) and the third particle then takes (0.03, 0.01). In figure 1(a), the dot in red indicates the optimal solution whereas figure 1(b) deduces the optimal solution if and only if the vectors of all the dots are combined. The block diagram indicates the operational steps required to compute the optimal solution using PSMSAO.

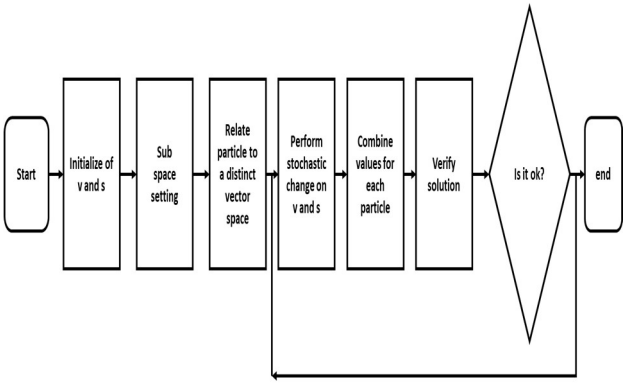

Figure 2. The operation involved in PSMSAO

4.2.2 The Proposed Algorithm (Primitive form)

Algorithm $PSMSAO(Dim)$
Given;

Input :=Objective function ***F*** and number of iteration ***iter***
Output :=best global position ***gbest***
The local best is given by ***pbest***

$$M = \begin{bmatrix} \dots, 3, 2, 1, 0, -1, -2, -3, \dots \\ \dots, 0.3, 0.2, 0.1, 0, -0.1, -0.2, -0.3, \dots \\ \dots, 0.03, 0.02, 0.01, 0, -0.01, -0.02, -0.03, \dots \\ \vdots \end{bmatrix},$$

where $mij = \frac{k}{10^{i-1}}$, $k \in \mathbb{Z}$, P x D, there can be many M with different ranges such as $M_1, M_2, .., M_n$

$$\text{Pbest} = \begin{bmatrix} m_{0k}, m_{0k}, .. \\ m_{1k}, m_{1k}, .. \\ m_{2k}, m_{2k}, .. \\ \vdots \end{bmatrix}, \text{ P x D}$$

P is the level of precision and D is the element in range of the boundary
P = dim
Start
For all i of M_1, M_2, M_3,.. M_n
for each element '*a*' in range D in M**:**
for each element '*b*' in range D in M_1**:**
for each element '*c*' in range D in M_2**:**
⋮
for each element '*q*' in range D in M_n**:**
if f(a, b, c,…,q) < f(pbest [i])
pbest [i] = [a, b, c,..,q]
gbest = pbest [1] $+ \sum_{a=2}^{m}$ pbest [a]
End

From Fig. 2, the flowchart considers the flow of operation from start to end. The—initialization parameters denoted as velocity v and position s vectors prior to the search space settings. The third operational block in Fig. 2 is the assigning of the particles into distinct vector spaces. Then the random motion of the particles by a stochastic approach ensures the update of the positions' vectors of each particle towards the optimal solution, such that each position's vector of the whole population behaves as a decimal place value of the optimal solution within the search spaces. The solution is attained by making the current sum the global solution if and only if it is less than the global solution, of a minimization problem, if not then it goes back to perform stochastic perturbation on v and s then continues the verification of the solution as optimal or not. If the solution is verified as the optimal, then the process ends. The algorithm above is a primitive form with no stochastic operations giving the algorithm a guaranteed convergence in real time (this can easily be verified by implemetation).

5. Numerical Experiments

In the experiments, the proposed PSMSAO, PSO, Artificial Bee Colony (ABC), Bat algorithm, Genetic Algorithm (GA) and Simulated Annealing (SA) algorithms are tested with the CEC2022 test functions.

5.1 Experimental Setup

Table 1
The parameter settings for the various algorithms

Algorithm	Parameters and values
PSO	number of particles: 25, maximum iteration: 250, c1 = 1.2, c2 = 1.225
ABC	Number of particles: 30, maximum iteration: 250, employed bees percentage: 0.5
Bat	Number of particles: 40, maximum iteration: 250, fixed loudness and rate: 0.7
GA	Alpha: 0.75, beta: 0.25, mutation rate: 0.1, crossing-over rate: 0.9, number of particles: 50, maximum iteration: 250
SA	P1: 0.7, Pn: 0.001, number of accepted solutions: 0.0, maximum iteration: 250
PSMSAO	C1: 0.12, c2: 1.2, maximum iteration: 100, number of particles: 3

The maximum iteration and number of particles of PSMSAO are intentionally left lower than the rest of the other algorithms. The machine specifications are 4.00GB RAM, 2.66GHz Intel duo core CPU and platform is windows 10, 64-bit operating system.

Both algorithms are run on 900 iterations for Table 4 and 5. The mean, variance and least error possible (LEP) are obtained from 100 samples with 10 function evaluations for each algorithm. From the data presented in table 5, PSMSAO gives the least error recordable.

The performance per means and standard deviations are outlined in Tables 3 and 4, the data illustrates that PSMSAO is more accurate in attaining the optimal solution per the experiment.

5.2 Application to Real-World Problems

The Tension-Compression String Problem *(TCS)*

Mathematical model of TCS:

Consider $X = [x_1, x_2, x_3] = [d, D, P]$

Minimize $F(x) = (x_3 + 2)x_2x_1^2$

Subject to

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0$$

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3)} + \frac{1}{5108x_1^2} - 1 \leq 0$$

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

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

With

$0.05 \leq x1 \leq 2, 0.25 \leq x2 \leq 1.3$ and $2 \leq x3 \leq 15$.

Table 14. Statistical results of optimization algorithms for the tension/compression spring design problem

	psmsao	goa	sabo	pso
mean	0.01	0.01206	0.012665	2.21E+13
best	0.01	0.01206	0.012665	0.017729
worst	0.01	0.01206	0.012665	3.93E+14
std	1.1E-18	7.63E-18	1.3E-18	9.75E+13
median	0.01	0.01206	0.012665	0.017729

The Quadrotor Drone Optimization Problem

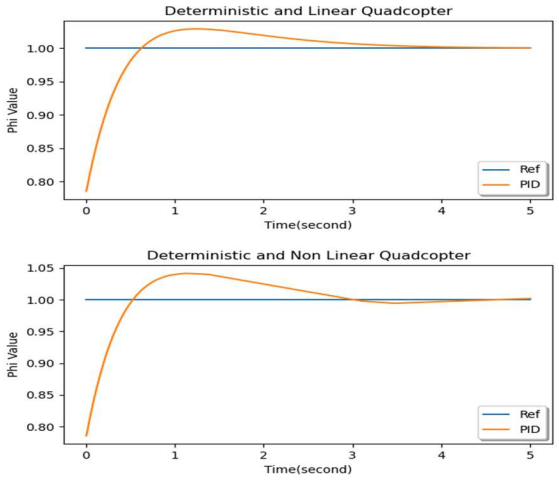

Figure 3. The quadrotor performance before optimization

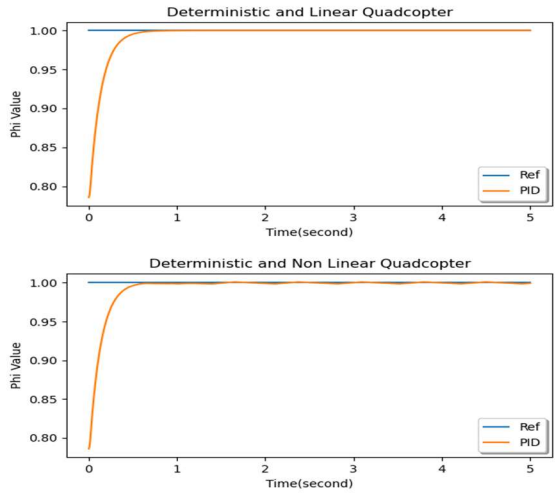

Figure 4. PSMSAO optimization of quadrotor performance

5.2 Discussion

The improved performance of PSMSAO over PSO is based on the facts that, the search space of PSMSAO is much smaller than PSO, since PSMSAO particles operate over discrete finite search unlike PSO with continuous discrete, and a much fewer particle number in PSMSAO are required for estimating optimal solution. In addition, PSMSAO particles are required to focus on only a portion of the solution such that their combined effect would result in the optimal solution; hence there is a level of synergy between the particle populations.

6 Conclusion

This paper provides a more computationally efficient way of finding the optimal solution of functional optimization problems as depicted in Section 5, in so doing the proposed algorithm constrains the particles to operate in a discrete search space and utilizes a relatively smaller particle size in its operation, Table 1 depicts the particle size for all the algorithms used in the experimentation. The proposed algorithm is able to find the optimal solution in the shortest possible time with minimal resources or particles and computational space. The proposed algorithm may be considered a PSO variant with strategies which have proven to optimize the search operation in estimating the optimal solution without much computational efforts. The synergic nature of the particle's operation under the proposed algorithm is paramount in estimating the expected optimal solution in most cases of the experimentation, as each particle is required to perform a single operation only, which is estimating the decimal place values that makes up the solution for the optimal solution. The knowledge of the chaotic nature and motion of the particles all at once in the search space shows that each particle does some work vehemently in estimating the optimal solution whereby most particle operations are redundant. Therefore, there is the necessity of each particle to operate in synergy in estimating the global optimal solution where each particle is in prior use and none results in redundancy.

As a future work, the proposed algorithm can be extended for hybridization with other metaheuristic algorithms for adequate fine tuning.

Table 3
Performance Evaluation on CEC2022 for 10 dimensions

F		ABC	PSO	GA	SA	PSMSAO	BA
Rosenbrock	Best	14429612957.012	3.40150	18.09908	129681.0	**0.0**	0.00331
	mean+std	100317677191.50473+87056926033.4411	55.34598119 + 29.30277	37.8428 +18.09908	129681.0+0.0	**0.0+0.0**	82.499 + 315.239
	Worst	590895620189.9498	157.6641	177.7700317	129681.0	**0.0**	2576.08
escaffer6	Best	0.000320	0.016180	1.013127	0.16434298	**0.0**	0.038866
	mean+std	0.5726 + 0.5363	0.518916 + 0.51278	1.18480 + 0.4312	0.574600 +0.20824	**0.0+0.0**	0.279538+ 0.163387
	Worst	2.280437	2.28944	2.6188	1.09739	**0.0**	0.86440417
happycat_func	Best	1817.312	0.032025	0.128315	3.427736	**0.0**	1.20250
	mean+std	3651.45420 + 913.77	0.135736 + 0.068072	0.64175 + 0.312256	5.672281	**0.0+0.0**	5.7605 + 2.15180

					+2.03703		
	Worst	6986.352	0.37883	1.425461	7.99963	**0.0**	11.4448
hgbat_func	Best	40810.78	0.08088	0.028355	47.14061	**0.0**	4.32123
	mean+std	71203.133 + 17432.35	0.236786 + 0.078848	1.7821569 + 2.062914	70.8690 +30.3075	**0.0+0.0**	35.62147 + 16.1711
	Worst	135501.8	0.387021	6.976929	159.4193	**0.0**	83.79283
schaffer_F7_func	Best	846780.5	0.125702	0.322020	6.04975	**0.0**	2.13110
	mean+std	29298165.795 + 56040125.377	0.370711 + 0.166273	0.8683535+0.223889	8.46897+0.744342	**0.0+0.0**	5.985149 + 1.53733
	Worst	432570682.989	0.84476	1.473090	9.44290	**0.0**	9.78343
Levy_func	Best	0.000216	0.005386	0.200403	0.06768	**1.499e-32**	7.883053e-09
	mean+std	0.119 + 0.4002	0.12171 + 0.06738	0.503877+0.12744	0.205865+0.0824	**1.499e-32+0.0**	2.23017 +1.77730
	Worst	3.01443	0.320002	0.83240	0.41699	**1.499e-32**	7.18168
zakharov_func	Best	0.006769	0.139314	1.00989	1.791229	**0.0**	2.873e-08
	mean+std	109.02 + 391.588	1.1214 + 0.643012	5.007707+1.9607	11.80224+16.6967	**0.0+0.0**	2.42415 +6.18105
	Worst	3863.682	3.23885	9.64616	93.90911	**0.0**	36.5299

Table 4
Performance Evaluation on CEC2022 for 20 dimensions

F		ABC	PSO	GA	SA	PSMSAO	BA
Rosenbrock	Best	271933793191.38635	51.11174	183.8586	273771.0	**0.0**	435222.803061
	mean+std	1184734626822.5166 + 812505768036.5149	359.0471 + 162.7657	688.8477 + 191.2155	273771.0 + 0.0	**0.0+0.0**	1433457.34 + 442175.519
	Worst	5398239946650.859	1210.286	1374.73	273771.0	**0.0**	2392841.97
escaffer6	Best	0.013707	0.797922	2.02625	0.944125	**0.0**	0.58206
	mean+std	1.19300 + 1.130699	2.42790266 + 1.01187	2.143022 + 0.401905	2.239909+ 0.51930	**0.0+0.0**	1.57746 +0.48161
	Worst	4.922269	5.12206	3.962735	3.488284	**0.0**	2.67039
	Best	5548.406	0.044796	0.440203	4.013609	**0.0**	7.795812

happycat_func	mean+std	8285.886 + 1526.809	0.188261 + 0.08037	1.516705 + 0.41577	5.93270 + 2.060727	**0.0+0.0**	13.1588 +1.97682
	Worst	12364.87	0.460192	2.441011	8.661791	**0.0**	17.4190
hgbat_func	Best	210870.5	0.049961	6.848806	101.7324	**0.0**	224.6894
	mean+std	367508.2823 + 80826.9860	0.161226+0.0524	17.4034 + 3.981831	123.1707+20.8884	**0.0+0.0**	384.54698 + 63.9213
	Worst	641120.4848	0.295616	28.94430	314.3386	**0.0**	539.9542
schaffer_F7_func	Best	32417713421.0963	0.260751	0.92506	3.002141	**0.0**	7.31186
	mean+std	21222833547629.203+ 45891968128604.16	0.531294 + 0.115987	1.56423 + 0.240677	9.274485+1.34065	**0.0+0.0**	10.7616 +1.01684
	Worst	28304859116681.1	0.875588	2.123187	9.925722	**0.0**	12.59636
Levy_func	Best	0.004243	0.152296	1.07533	0.406523	**1.499759e-32**	5.12202
	mean+std	2.005690 + 5.59039	0.435998 + 0.122276	1.935079 + 0.41493	0.680487+0.14348	**1.499759e-32+0.0**	16.14791 + 7.148505
	Worst	46.50500	0.875453	3.439286	1.136120	**1.499759e-32**	41.65232
zakharov_func	Best	0.011631	1.339727	11.26493	6.357626	**0.0**	15.29668
	mean+std	197.3305 + 202.8956	5.68379 + 2.384073	21.06020+4.98271	574.4499+ 1134.604	**0.0+0.0**	373.2710 + 268.6219
	Worst	875.0923	14.72379	38.53057	7942.358	**0.0**	1217.1415285

7. References

Agrafiotis, D. K., & Cedeño, W. (2002). Feature selection for structure-activity correlation using binary particle swarms. In *Journal of Medicinal Chemistry*. (pp. 1098–1107). PubMed.

Arumugam M. S., & Rao M. V. C. (2006).On The Performance of the Particle Swarm Optimization Algorithm with various Inertia Weight variants for Computing Optimal Control of a Class of Hybrid Systems, In *Discrete Dynamics in Nature and Society, Hindawi Publishing Company, DOI 10.1155/DDNS/2006/79295*. Vol. 2006. (pp. 1 – 17).

Chen, W. N., Zhang, J., Chung, H. S. H, Zhong, W. L., Wu, W. G., & Shi, Y. H. (2010). A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems, In *IEEE Transactions on Evolutionary Computation*, Vol. 14. No. 2. (pp. 4460 - 4493).

Hendtlass, T. (2005). WoSP: A Multi-Optima Particle Swarm Algorithm, In *2005 IEEE Congress on Evolutionary Computation* (pp. 727 - 734). Ieee.

Heppner, H., & Grenander, U. (1990). A stochastic non-linear model for coordinated bird flocks. In *S. Krasner (Ed.), The ubiquity of chaos* (pp. 233–238). Washington: AAAS.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (pp. 1942 – 1948). Ieee.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of the conference on systems, man, and cybernetics.* (pp. 4104–4109). Piscataway: Ieee.

Lalwani S., Kumar R., & Gupta N. (2013). A Preview on Particle Swarm Optimization variants and their Applications to Multiple Sequence Alignment, In *Journal of Applied Mathematics and Bioinformatics*, Vol. 3. No. 2. (pp. 87 – 124).

Liu,W.,Wang, Z., Yuan, Y., Zeng, N., Hone, K., & Liu, X., (2021) A Novel Sigmoid-Function-Based Adaptive Weighted Particle Swarm Optimizer, In *IEEE Transactions on Cybernetics*, Vol. 51, (pp. 1085 – 1093).

Montes de Oca M. A., Stutzle T. & Birattari M. (2009). Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm. In *IEEE Transactions on Evolutionary Computation*, Vol. 13. No. 5. (pp. 1120 - 1132). Ieee.

Parsopoulos, K. E., & Vrahatis, M. N. (2001). Particle swarm optimizer in noisy and continuously changing environments. In *M. H. Hamza (Ed.), Artificial intelligence and soft computing* (pp. 289–294). Anaheim: IASTED/ACTA.

Poli, R., Kennedy, J. & Blackwell, T. (2007), Particle Swarm Optimization: An Overview, In *Swarm Intell, Springer Science, Business Media*, (pp. 33 – 57).

Pugh, J., Martinoli, A., & Zhang, Y. (2005). Particle swarm optimization for unsupervised robotic learning. *In Proceedings of IEEE swarm intelligence symposium (SIS).* (pp. 92–99). Piscataway: Ieee.

Shi, Y., Liu, H., Gao, L., & Zhang, G. (2011). Cellular Particle Swarm Optimization, In *Inform, Sci, doi: 10.1016/j.ins.2010.05.025.* Vol. 181. No. 10. (pp. 4460 - 4493) DBLP.

Valle, Y. D., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C. & Harley R. G. (2008), Particles Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems, In *IEEE Transactions on Evolutionary Computation*. Vol. 12. No. 2. (pp. 171 – 195). Ieee.

Wei, C., He, Z., Zhang, Y., & Pei, W. (2002). Swarm directions embedded in fast evolutionary programming. In *Proceedings of the IEEE congress on evolutionary computation (CEC)* (pp. 1278–1283). Honolulu, HI. Piscataway: IEEE.

Zhang, H, Yuan, M. & Liang, Y. and Liao, Q. (2018). A novel particle swarm optimization based on prey–predator relationship, In *Applied Soft Comput*ing, vol. 68, (pp. 202–218). DOI:10.1016/j.asoc.2018.04.008

Zhao, S. Z., Liang, J., Suganthan, P. N., & Tasgetiren, M. F. (2008). Dynamics Multi-search Particle Swarm Optimizer with Local Search for large Scale Global Optimization, In *IEEE Congress on Evolutionary Computation, doi: 10.1109/CEC.2008.4631320*. (pp. 3845 - 3852). DBLP.

Zhou, H., & Wei. X., (2021) Particle Swarm Optimization Based on a Novel Evaluation of Diversity.In *Intelligent Manufacturing and Information Engineering*. vol. 14. (pp.202-218). MDPI.

Zhu, Z., Zhong, T., Wu, C., & Xue, B. (2022). Dynamic Multi-search Particle Swarm Optimization with Center Learning Strategy. In *Applied Soft Comput*ing (pp. 141–147).