# Name: Derrick Donkor

# Email: [DerrickDonkorsci@gmail.com](mailto:DerrickDonkorsci@gmail.com)

# **THE PARTICLE SWARM MULTI-SUBSPACE AUGMENTATION OPTIMIZATION**

# **ABSTRACT**

In this article, the proposed Particle Swarm Optimization (PSO) variant uses a search space with distinct subspaces for each particle in the population respectively 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 a particle to a discrete subspace and combines each particles position to form the solution at every functional evaluation. This assignment of the particle to a discrete subspace is suitable for swarm size reduction and quick convergence with less iteration. The theoretical basis is provided for the proposed algorithm and empirical studies are conducted to compare the proposed algorithm with PSO and some selected optimization algorithms on reference benchmark test functions. Based on the experimental results, the proposed algorithm has a higher accuracy than the other optimization algorithms.

Key words: Particle swarm optimization, Subspace, fast convergence and global solution.

## **1 Introduction**

Ever since its introduction in 1995 by [1], PSO has emerged as one of the most popular optimization algorithms. From [10] PSO was influenced by Heppner Grenander’s work [11] 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, adding up to the design implementation 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. Convectional optimizers are generally incapable of attaining highly accurate results under less iteration. 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. PSO is designed such that the subspaces within which each particle operates are of the same size and has a cardinality of that of a real number, the elegance in the design of the proposed algorithm is that it enables the swarm particles to work in a more synergic and harmonious way such that the subspaces for each of the particles span over a discrete set of integers and that the overall result of the solution is composed of each and every particle position in respective subspaces.

This section presents some available variants on PSO and outlines their key features and differences from the proposed algorithm, the section records available literature from the time of its creation, of PSO and variant, to date.

The first introduction of the Particle Swarm Algorithm came from [8], 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 [12] which was in the form of Binary Particle Swarm, as the earliest of its kind, the algorithm 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.

[13] proposed a roulette wheel base 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.

[16] successfully embedded velocity information in an evolutionary algorithm. This was done by replacing a version of PSO velocity in a fast evolutionary programming (FEP) algorithm with Cauchy mutation to give the population direction. Their published results indicate that the approach is very successful on a range of functions; the new algorithm found global optima in tens of iterations, compared to thousands for the FEP versions tested.

[2] introduced Cellular Particle Swarm Optimization. In this work, a mechanism of cellular automata is applied to the velocity update of Particle Swarm Optimization to avoid the position vector from being trapped in a local optimum.

[17] proposed a novel bio-inspired prey-predator PSO variant, there is prey catching, escaping and breeding where particles with less fitness can be removed from swarm or replaced. It introduces proportional-integral controllers to ensure population diversity.

[18] 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.

[20] proposed a dynamic multi-swarm 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.

[19] introduce an acceleration update strategy which utilizes sigmoid-function based weighting rules. The algorithm combines both the global best position and the personal best position to decide on how to update the weights. It outperforms the PSO variants in its study and has an enhanced convergence rate than the PSO algorithm.

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.

The contributions of the proposed paper are as follows;

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

2. To present a PSO based technique whereby there is a more harmonious and a synergic relationship between particle operations.

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. PSO-based optimization techniques usually do not implement the particle swarm algorithm to mimic a harmonious, less chaotic societal living in general. So, 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 organization of search-spaces and a mechanism of 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-dimensional vector 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.

**2.3 Overview of the Proposed Algorithm**

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 PSSAO and PSO is how particles coordinate to find the optimal solution. The problem search space of PSO is made up of a real number d- dimensional vector subspace of a given span, where d is the size of the dimension. On the other hand, PSSAO has particles with each having a discrete number d-dimensional vector subspace, where each particle's optimal location corresponds to a particular decimal place value of the overall optimal solution. For instance; if an optimal solution is found to be , 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 , the second a value of and the third a value of . From the example above there are 3 particles with 1-dimensional integer vector subspace, and each particle generally contributes to the formation of the optimal solution.

*Lemma 1: Given a real number set and a discrete number set of the same span, the cardinality of the real number set is always greater than that of the discrete number set.*

Lemma 1 is true simply because a real number set is a continuous set of numbers while a discrete number set is primarily discontinuous. An instance of the discrete number set is a set of integer numbers.

## **3 The proposed Algorithm**

The optimizer proposed has its performance evaluated by functional optimization approaches, specifically with bench test functions. Detailed algorithmic and block diagram description is presented in Sections 3.4.1 and 3.4.2 respectively.

Although some aspects like velocity update, position update, local best and global best estimation procedure are utilized from PSO, this concept is not merely about just a couple of particle objects, where each particle tries to stochastically find the global solution it is about making each particle object span a single place value range, hence, some benchmark functions were used for the comparison between this proposed concept and other optimization techniques. The comparative analysis uses constant values, such as acceleration and inertia, for both PSO and PSSAO presented in table 1. 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 PSSAO and PSO under a couple of dimensions.

**4.1 Mathematical Formulation of the Proposed Algorithm**

Given a real number set of a given range denoted as ℝ and a discrete number set of same range denoted as ⅅ.

In other words,

} (1)

[ , ], (2)

Given that is the minimum value, is the maximum value and is the discrete interval between the elements of .

From lemma 1;

(3)

The cardinality of a real number set of a given size is greater than a discrete set of the same size.

Since the Particle entities of PSO has a vector space for its position and velocity as , n-dimension and that of PSSAO has a vector space of , n-dimension for its position and velocity.

Given that is the velocity of particle at iteration

, are the acceleration constants

is the inertia constant, is the random number at iteration

are local best, global best and position respectively

(4)

(5)

The particle object used in PSO and PSSAO is denoted by ;

is the set of all particles such that;

(6)

(7)

Equation (7) coordinates the iteration number with and over a random probability distribution. For PSO the span for has a continuous range whilst that for PSSAO is discrete in nature.

From PSSAO, the optimal solution at the iteration is given by;

(8)

(9)

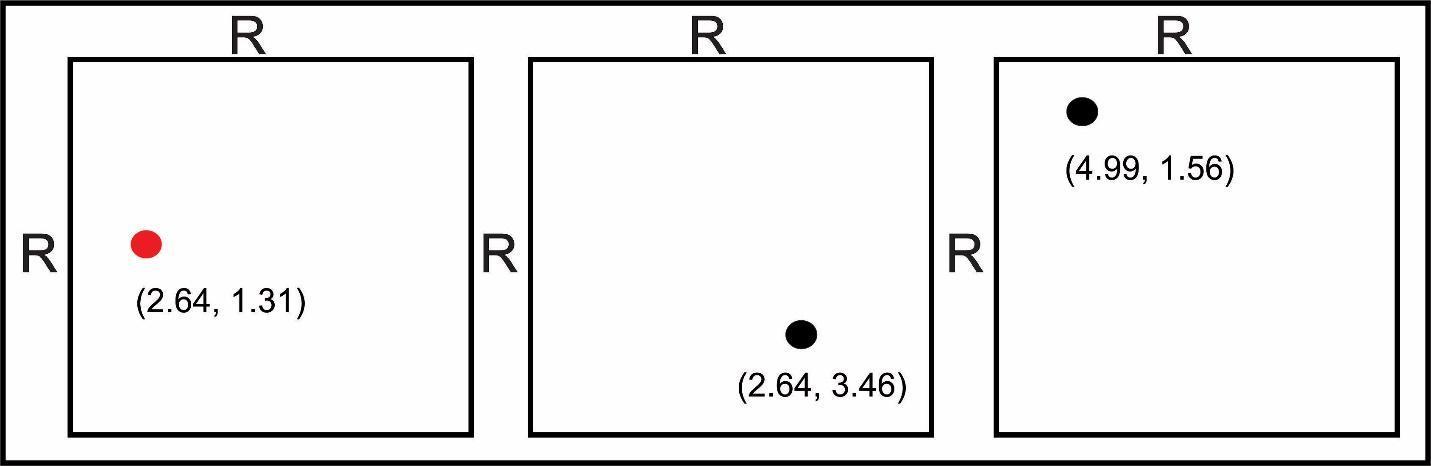
Equation 8 is that which combines the separate position values of all the particles in their respective subspaces.

The span for each particle is given by;

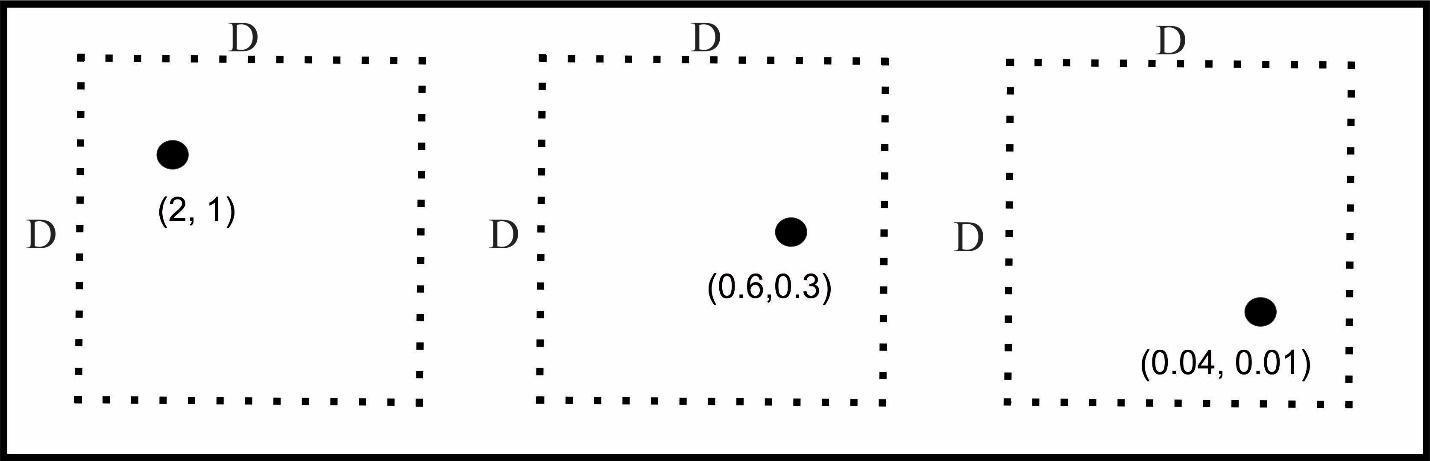
(10)

**4.2 Conceptual View of PSO and PSSAO**

**4.2.1 Simple Representation of PSSAO optimal solution Discovery Mechanism**



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



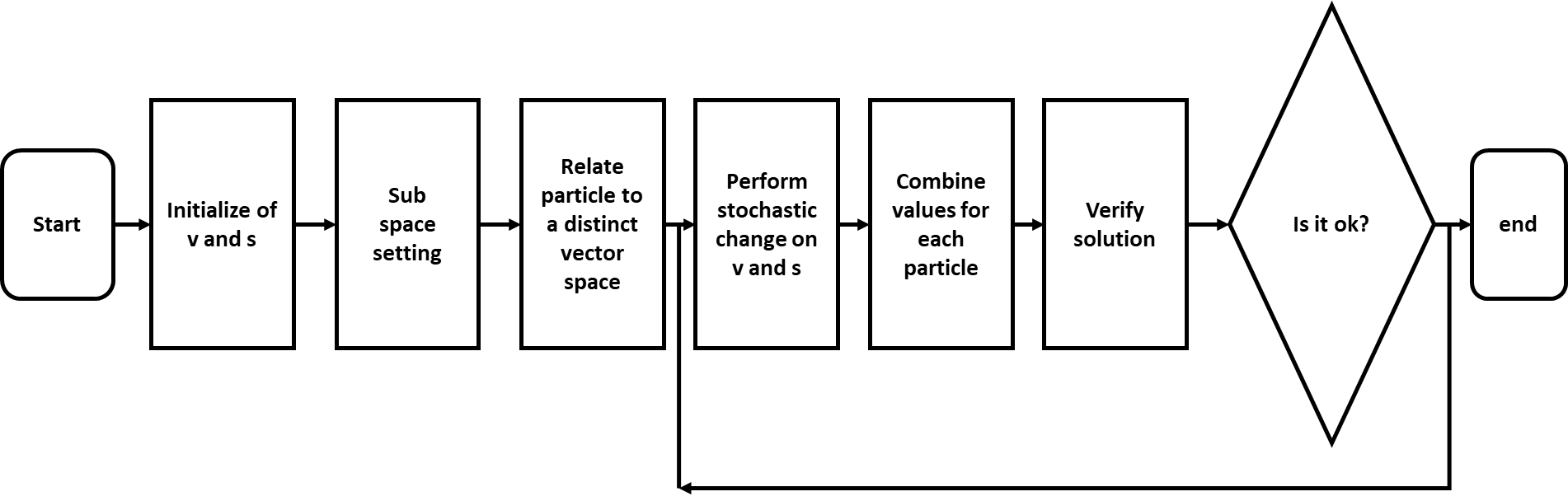
**Fig. 1(b).** The representation of PSSAO optimal solution search activity

Figure 1(a) and 1(b) presents the design concept into a logical pictorial diagram.

The optimal solution is (2.63, 131) in both Figure 1(a) and 1(b); there are three particles in both figure 1(a) and 1(b). For every 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 PSSAO.



**Fig. 2.** The vital operation of the process involved in PSSAO

**4.2.2 The Proposed Algorithm**

**Algorithm** *PSSAO*

Given;

*Input* Objective function ***F***, particles population ***P*** (made up of position ***S*** and velocity ***V*** as array properties) and number of iteration ***iter***

*Output* best global position ***gbest***

Decimal place setting function is ***decset*** (a, n); where ‘a’ is the particle list with ‘n’ locations in ***P***

The local best is given by ***pbest***

**Start**

1. Initialize vector ***S*** and ***V*** to population size, and also ***pbest, c1,c2,Vk*** and ***Sk***
2. Loop1: for each step in iteration
3. Loop2: for each jth element in population *P*
4. Loop3: for each ith decimal places
5. Compute value of array ***S*** with position of members as decimal places by decset(S,n)
6. Compute value of ***pbest*** with position of members as decimal places by decset(pbest, n)
7. If F (decset(*S*, *n*)) < F(decset(***pbest***, *n*)) set ***pbest***[i] = *S*[i]
8. End loop3
9. Set ***gbest*** = ***pbest***
10. Compute random number in [0, 1] as ***rand***
11. *V*[j] = *Vk*[j] + *c1*\*rand\*(decset(*pbest*,*n*) - decset(*Sk*,*n*)) + *c2*\*rand\*(decset(*gbest*,*n*) - decset(*Sk*,*n*))
12. *Vk*[*j*] = *V*[*j*], *S*[*j*] = *Sk*[*j*] + *V*[*j*] and *Sk*[*j*] = *S*[*j*] as integer
13. End Loop2
14. End Loop1

**End**

From Fig. 2, the flowchart considers the flow of operation from start to end. Theinitialization parameters denoted as velocity and position vectors prior to the subspace 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 position vectors of each particle towards the optimal solution, such that each position vector of the whole population behaves as a decimal place value of the optimal solution within the subspaces. The solution is verified by making the current synergic 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 and then continues the verification of the solution as optimal or not. If the solution is verified as the optimal, then the process ends.

## **5 Numerical Experiments**

In the experiments, the proposed PSSAO, 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 = 0.075, c2 = 0.225 |
| 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 |
| PSSAO | C1: 0.12, c2: 1.2, maximum iteration: 100, number of particles: 3 |

The maximum iteration and number of particles of PSSAO 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. Table 2, 3 and 4 presents the most optimal solution attainable by estimation under 9, 100 and 10000 iterations respectively for both PSO and PSSAO algorithm.

Table 2 Comparison of PSO and PSSAO under 9 iterations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Function | PSO Optimal Solutions | PSO Optimal Point | PSSAO Optimal Solutions | PSSAO Optimal point | Actual optimal solution | Actual optimal point |
| Ackley | [0.453, 0.247] | 2.132 | [0.0, 0.0] | 0.0 | [0, 0] | 0 |
| Griewank | [0.256, 0.0035] | 0.032 | [0.0, 0.0] | 0.0 | [0,0] | 0 |
| Rosenbrock | [0.685, 0.469] | 0.09 | [1, 1] | 0.0 | [1, 1] | 0 |
| Sphere | [0.035, 0.321] | 0.1 | [0.0, 0.0] | 0.0 | [0, 0] | 0 |

Table 2 illustrates the comparison between PSO and PSSAO in terms of their optimal solution estimation under nine iterations. For the given test functions PSSAO is better in estimating the global optimal solution than the PSO.

Table 3 Comparison of PSO and PSSAO under 100 iterations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Function | PSO Optimal Solutions | PSO Optimal Point | PSSAO Optimal Solutions | PSSAO Optimal point | Actual optimal solution | Actual optimal point |
| Ackley | [0.195, 0.0601] | 2.499 | [0.0, 0.0] | 0.0 | [0, 0] | 0 |
| Griewank | [0.1355, 0.08488] | 0.0109 | [0.0, 0.0] | 0.0 | [0,0] | 0 |
| Rosenbrock | [0.809, 0.637] | 0.0657 | [1.0, 1.0] | 0.0 | [1, 1] | 0 |
| Sphere | [0.081, 0.095] | 0.015 | [0.0, 0.0] | 0.0 | [0, 0] | 0 |

Table 4 Comparison of PSO and PSSAO under 10000 iterations

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Function | PSO Optimal Solutions | PSO Optimal Point | PSSAO Optimal Solutions | PSSAO Optimal point | Actual optimal solution | Actual optimal point |
| Ackley | [0.03416, 0.00219] | 0.51219 | [0.0, 0.0] | 0.0 | [0, 0] | 0 |
| Griewank | [0.010733, 0.024538] | 0.000208 | [0.0, 0.0] | 0.0 | [0,0] | 0 |
| Rosenbrock | [0.7999, 0.6384] | 0.04022 | [1.0, 1.0] | 0.0 | [1, 1] | 0 |
| Sphere | [0.00879, 0.04013] | 0.001688 | [0.0, 0.0] | 0.0 | [0, 0] | 0 |

The values recorded in the tables 2, 3 and 4 are the expected values of the optimal solutions. From the tables above, the proposed algorithm PSSAO attains the best possible optimal solution, considering the test functions outlined.

Table 5 Performance Measure on Various Dimensions

The benchmark functions are evaluated under 2, 5, and 10 dimensions. For all cases, the mean fitness scores of the best particles are estimated.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | PSO | | PSSAO | | Least error possible (LEP) | Technique that gave LEP |
| Test Function | Dimension | Mean | variance | Mean | variance |
| Ackley | 2 | 0.7126624 | 0.02246899 | 0.106202 | 0.0246570 | 0 | PSSAO |
| 5 | 1.92955601 | 0.02405623 | 1.265991 | 0.0549063 | 0 | PSSAO |
| 10 | 2.58782884 | 0.01944704 | 2.177654 | 0.0535811 | 0.4334990 | PSSAO |
| Griewank | 2 | 0.00139719 | 0.00015437 | 3.228e-8 | 3.2119e-8 | 0 | PSSAO |
| 5 | 0.03485342 | 0.00148153 | 0.000939 | 0.0001382 | 0 | PSSAO |
| 10 | 0.10780683 | 0.00227347 | 0.022811 | 0.0029359 | 0.0008301 | PSSAO |
| Rosenbrock | 2 | 0.02178860 | 0.00202041 | 0.002103 | 0.0007397 | 0 | PSSAO |
| 5 | 5.17941363 | 0.23402827 | 2.846445 | 0.1484891 | 0.0175966 | PSSAO |
| 10 | 49.0080491 | 0.90709210 | 37.71140 | 2.6978230 | 0.223917 | PSSAO |
| Sphere | 2 | 0.00371462 | 0.00031218 | 1.3019e-33 | 1.2954e-33 | 0 | PSSAO |
| 5 | 0.18587038 | 0.00726737 | 0.005665 | 0.0014999 | 0 | PSSAO |
| 10 | 0.97167471 | 0.02390041 | 0.296344 | 0.0378893 | 0.000004 | PSSAO |

Table 5 presents the performance measure on 2, 5 and 10 dimensions. Both algorithms are run on 900 iterations for Table 2, 3, 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, PSSAO gives the least error recordable.

**Table 6**

Performance Evaluation on CEC2022 for 10 dimensions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| F |  | ABC | PSO | GA | SA | PSSAO | 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 +2.03703 | **0.0+0.0** | 5.7605 + 2.15180 |
| 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 7**

Performance Evaluation on CEC2022 for 20 dimensions

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| F |  | ABC | PSO | GA | SA | PSSAO | 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 |
| happycat\_func | best | 5548.406 | 0.044796 | 0.440203 | 4.013609 | **0.0** | 7.795812 |
| 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 |

The performance per means and standard deviations are outlined in Tables 5 and 6, the data illustrates that PSSAO is more accurate in attaining the optimal solution per the experiment.

**5.2 Discussion**

The improved performance of PSSAO over PSO is based on the facts that, the search space of PSSAO is much smaller than PSO, since PSSAO particles operate over discrete subspace unlike PSO, and a much fewer particle number in PSSAO are required for estimating optimal solution. In addition, PSSAO 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. Also, modification of the proposed algorithm to solve different classes of combinatorial optimization problems is necessary for exploring other key aspects of the proposed algorithm.

**References**

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

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

[3] Chen, W. N., Zhang, J., Chung, H. S. H, Zhong, W. L., Wu, W. G., & Shi, Y. H. (April, 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).

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

[5] Lalwani S., Kumar R., & Gupta N. (June, 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).

[6] Arumugam M. S., & Rao M. V. C. (June, 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).

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

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

[9] Valle, Y. D., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C. & Harley R. G. (2008, April), 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.

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

[11] 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.

[12] 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.

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

[14] Parsopoulos, K. E., & Vrahatis, M. N. (2001, January). 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.

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

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

[17] Zhang, H, Yuan, M. & Liang, Y. and Liao, Q. (2018, April). 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](http://dx.doi.org/10.1016/j.asoc.2018.04.008)

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

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

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