Comparison of PSO and CLPSO Algorithms on Single Objective Real-Parameter Numerical Optimization

Yuhao Zhou[1a]

***Abstract*—This paper presents the progress of completing the first assignment of EE6227. Two algorithms were chosen here, the basic particle swarm optimization (PSO) and the comprehensive learning PSO (CLPSO). Using different parameters based on tuning experiments, the overall performance of these two algorithms on 10 selected problems including the Ackley's, Griewangk, and Weierstrass function would be compared and discussed. During the parameter tuning progress, statistical testing would be introduced to select the optimum parameters. Finally, the result of the convergence performance and variance of each algorithm would be presented.**

***Keywords*—Particle Swarm Optimization (PSO), Composition benchmark functions, global optimization, Comprehensive learning particle swarm optimizer (CLPSO), global numerical optimization.**

I. Introduction of Algorithms

Focused on single objective real-parameter optimization, in this assignment, the Particle Swarm Optimization (PSO) algorithm [1][2] and Comprehensive Learning Particle Swarm Optimization (CLPSO) [3] were utilized.

A. Particle Swarm Optimization

As a biological-inspired algorithm, the PSO algorithm simulates the predation behavior of birds in the natural environment which is conceptually simple and easy to implement. Compared with conventional Evolutionary Algorithms (EA), the PSO algorithm shows higher efficiency. Figure. 1. shows the fundamental procedure of PSO algorithms. Parameter w indicates the inertia weight, and c_1 c_2 indicates the acceleration constants, which would be tuned in the later sections.

The basic PSO algorithm flow can be concluded as follows:

1. Randomly initialize the particles at ith and generate their initial position $X_i = (X_i^1, X_i^2, \ldots, X_i^D)$ and velocity $V_i = (V_i^1, V_i^2, \ldots, V_i^D)$, then set the historical optimal $pbest_i = (pbest_i^1, pbest_i^2, \ldots, pbest_i^D)$ of the individual as the current position, and the best position discovered by the whole population is marked as $gbest = (gbest^1, gbest^2, \ldots, gbest^D)$.
2. Calculate the fitness value of each particle
3. If the historical optimal fitness value is better than the best position experienced in the group for each particle, then this historical optimal fitness value would be regarded as the current global optimal position

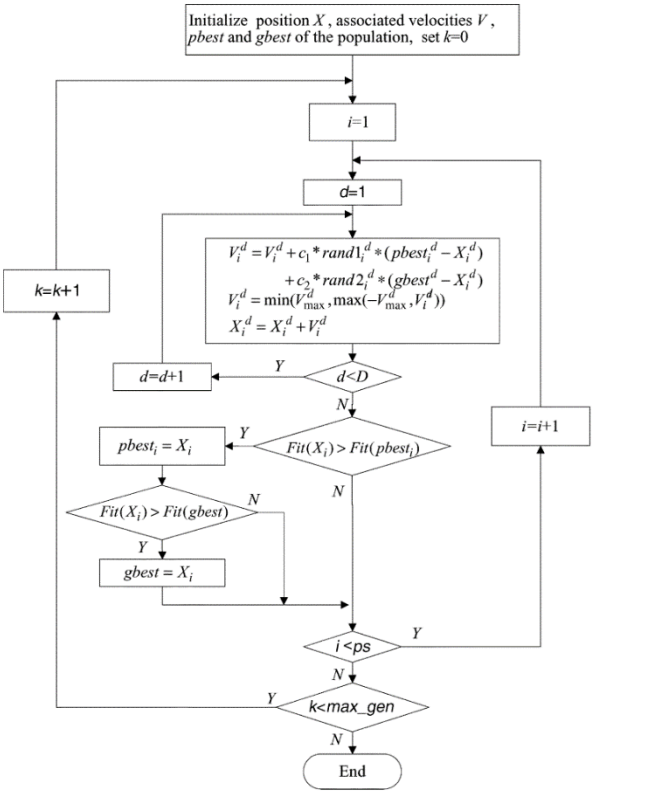


Fig. 1. Flowchart of the conventional PSO [3]

4. If the current fitness value is better than its historical optimal adaptive value at the updated best position for each particle, then this current position would be regarded as the optimal position
5. Using the following updating equation, the velocity of the particles is updated with two random numbers $rand1_i^d$ and $rand2_i^d$ in the range [0,1]. Refresh the d-th dimension of the i-th particle:

$$V_i^d \leftarrow V_i^d + c_1 * rand1_i * (pbest_i^d - X_i^d) + c_2 * rand2_i * (pbest^d - X_i^d)$$

$$X_i^d \leftarrow X_i^d + V_i^d$$

6. Continue to process if the termination condition is not satisfied; otherwise, break the loop. The termination condition can be set as a good adaptive value or reached the predefined iteration number

Vmax is the magnitude of particle's velocity on each dimension. Once if the $|V_i^d|$ is larger than V_{max}^d regulated, the

[a]Corresponding author: Yuhao Zhou {YZHOU035@e.ntu.edu.sg}
Matriculation Number: G2002124F
The author is with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639789 Singapore.

corresponding dimension's velocity would be assigned to $sign(|V_i^d|)\ V_{max}^d$.

In the later section on parameter tuning, the inertia weight w and acceleration constants c_1 c_2 would be the objective for analysis and discussion.

B. Comprehensive Learning Particle Swarm Optimization

One improvement on the PSO algorithm, the Comprehensive Learning PSO (CLPSO) algorithm was proposed by Liang et al. [3] in 2006. Three major improvements are listed as follows:

1. Using all particles' *pbest* to guide a particle's direction, rather than using on particle's *pbest* and swarm's *gbest* as the exemplars.
2. Different dimensions of a particle may learn from various virtual *pbests* within iterations
3. Instead of learning from two exemplars *pbest* and *gbest* in every generation, each dimension of a particle in CLPSO learns from just one comprehensive exemplar within regulated generations.

This method avoids the prematurity caused by *pbest* falling into the local optimum. Besides, the CLPSO algorithm featured with strong global search ability and strong robustness under the condition of sufficient iterations.

Fig. 2. shows the flow chart of the CLPSO algorithm. The improved velocity updating equation is as follows:

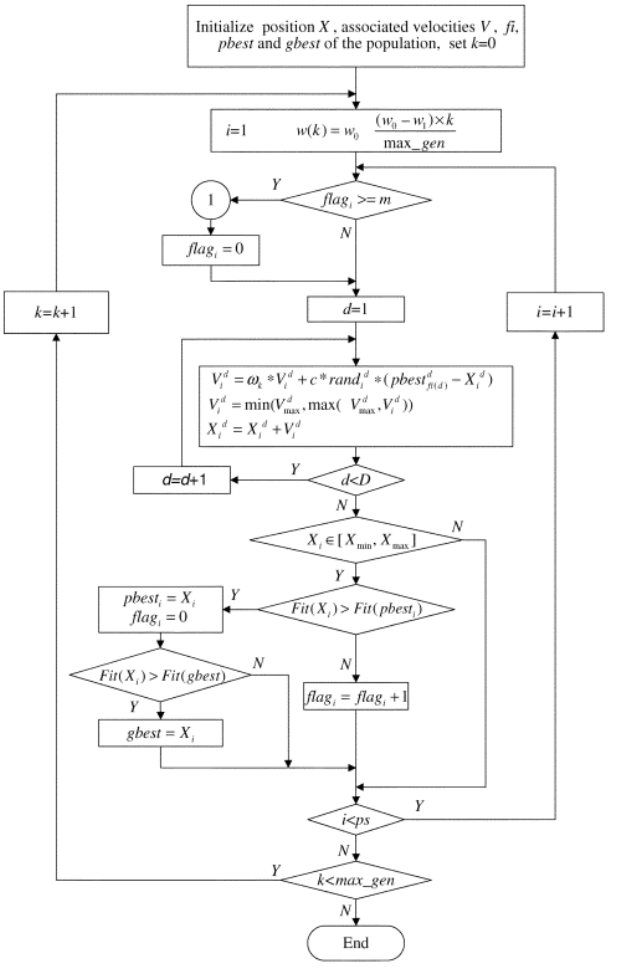
$$V_i^d \leftarrow w * V_i^d + c_1 * rand_d^i * (pbest_{fi(d)}^d - X_i^d)$$

$f_i = [f_i(1), f_i(2), \ldots, f_i(D)]$ denotes a set of particle indices according to each dimension of the particle i. $f_i(d)$ represents a comprehensive exemplar. The procedures of CLPSO could be summarized as follows:

1. Two particles randomly selected from the whole population which excludes the particle that had already updated velocity
2. Select the better particle according to the fitness value of their *pbest*
3. Employ the better *pbest* as the exemplar to learn from the correlated dimension. Randomly select one dimension of another particle's *pbest* if all exemplars of one particle are its own *pbest*.
4. Then new positions of *pbest* can be generated in the search space based on the previous procedure in updating various particles' optimal historical positions.

Similar to the PSO algorithm, there are several parameters in the CLPSO algorithm, including the inertia weight w_{max}, w_{min}, and acceleration constants c. Besides the above, the refreshing gap m was introduced. According to the previous research [3], better results of the convergence can be obtained with an m set at seven when solving single-objective problems.

In the later section on parameter tuning, to simplify the calculation, the refreshing gap m was set at seven for all the test functions employing the CLPSO algorithm. The parameter tuning would be mainly focused on two inertia weight factors and the acceleration constants.



(1) :Insert the flowchart in Fig. 1 here.

ps: population size
k: generation counter from 1 to *max_gen*
d: dimension
max_gen: maximum generations
i: particle's id counter from 1 to *ps*
X_i^d : i^{th} particle's d^{th} dimension's value

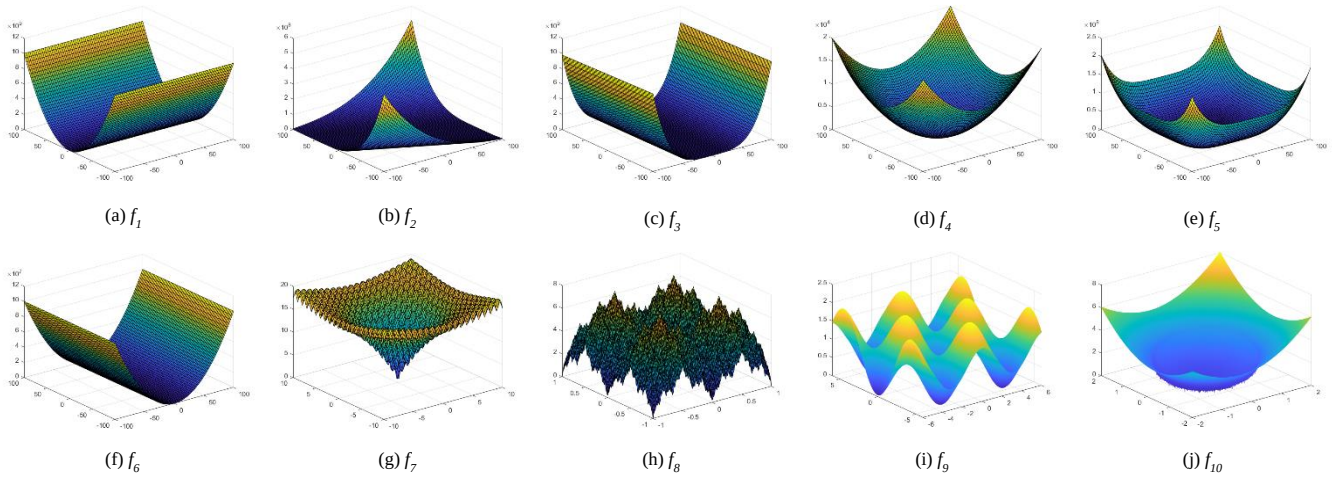
Fig. 2. Flowchart of the CLPSO [3]

C. Some other Variants PSO

Based on the PSO algorithms, many derive optimization methods had been proposed for the last decades. These optimized algorithms include the unified particle swarm optimizer (UPSO) [4], fitness-distance-ratio-based PSO (FDR-PSO) [5], cooperative particle swarm optimizer (CPSO-H) [6], and more. Various methods have specific improvements focusing on different real problems. Here the PSO and CLPSO algorithms were the focused ones which other algorithms would not be discussed.

TABLE I
THE TEN TEST FUNCTIONS SELECTED

f	name	test function	search space	target value
1	Cigar	$f_1 = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2$	$[-100,100]^D$	0
2	Zakharov	$f_2 = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5x_i^2)^2 + (\sum_{i=1}^{D} 0.5x_i^2)^4$	$[-100,100]^D$	0
3	Rosebrock	$f_3 = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	$[-100,100]^D$	0
4	Rastrigin	$f_4 = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-100,100]^D$	0
5	Schewfel	$f_5 = 418.9829 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-1000,1000]^D$	0
6	Discus	$f_6 = 10^6 x_1^2 + \sum_{i=2}^{D} x_i^2$	$[-100,100]^D$	0
7	Ackley's	$f_7 = 20 - 20e^{-0.2\sqrt{\frac{1}{n}*\sum x_i^2}} + e - e^{\frac{1}{n}\sum\cos(2\pi x_i)}$	$[-100,100]^D$	0
8	Weierstrass	$f_8 = \left(\sum_{k=0}^{kmax}\left(0.5^k \cos\left(2\pi \cdot 3^k (xi + 0.5)\right)\right)\right) - D\sum_{k=0}^{kmax}\left(0.5^k \cos(2\pi \cdot 3^k \cdot 0.5)\right)$	$[-100,100]^D$	0
9	Griewank's	$f_9 = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-100,100]^D$	0
10	Happy Cat	$f_{10} = \lvert\sum_{i=1}^{D} x_i^2 - D\rvert^{1/4} + (0.5\sum_{i=1}^{D} x_i^2 + \sum_{i=1}^{D} x_i)/D + 0.5$	$[-100,100]^D$	0



(a) f_1 (b) f_2 (c) f_3 (d) f_4 (e) f_5

(f) f_6 (g) f_7 (h) f_8 (i) f_9 (j) f_{10}

Fig. 3. Test function near the extreme point image

II. PSO AND CLPSO ALGORITHM EXPERIMENT

In this section, the procedure of verification would be introduced. The prerequisite and requirements are as follows:

Solving 10-dimensional (10D) problems using the two algorithms, the initial population size is set at 10 and the maximum evaluations (Fes) is set at 50000.

During the parameter tuning, the experiments can be conducted using 10 parameter tuning progress, repeat 10 times on 5 selected test functions.

After data analysis, the optimum parameters would be selected and then apply the optimum parameters in the algorithms to run all 10 test functions 30 times. Finally, the comparison of the variance between PSO and CLPSO algorithms including data and plots would be shown. And the conclusion would be made.

A. Test Functions

To test the PSO and CLPSO algorithms, initially, 10 test functions were utilized. Here all the test functions are derived from CEC 2017 bound-constrained benchmarks [7]. The name of the test functions and the value range of the variable are listed in Table I.

B. Parameter Tuning: Grid Search

First and foremost, the initial fundamental parameters were selected for both PSO and CLPSO algorithms as mentioned in the beginning of this section.

According to the requirement, the tuning experiment would be conducted only on 5 test problems, here problems f_1, f_2, f_3, f_4, f_5 were chosen. Then these functions would be conducted using 10 repetitions based on the grid search method to find the optimum parameter set for the above test problems with the best convergence performance.

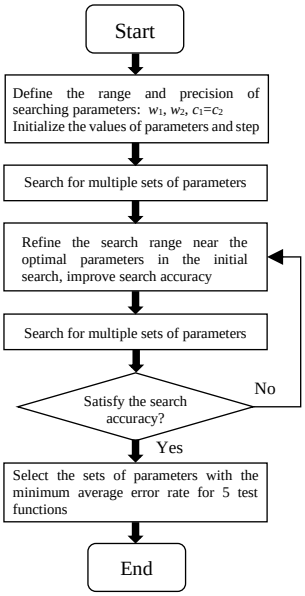


Fig. 3. Flowchart of the Grid Search Method

As shown in Fig. 3., for the PSO algorithm, the description of the parameter tuning procedure is as follows:

1. Determine the parameters that need to be adjusted. Here the inertia weight w_1, w_2, and acceleration constants c_1=c_2 are chosen
2. Set the grid search range of the parameters. In the initial grid search experiment, the search range of w_1 is [0.5, 1] with precision at 0.1. Similarly, the search range of w_2 is [0.1, 1] and the search range of c_1=c_2 is [1.4, 2], both precisions are 0.1
3. Run the grid search, each test function in each parameter set would calculate for 10 times (10 repetitions). After computing, export the data for initial analysis
4. For each parameter set, calculate the average error rate for all the 5 test functions. More specifically, in one parameter set, compare the mean *gbest* value with the target value of test functions, and then record the mean error rate corresponding to the parameter set
5. Narrow the search range and improve the search precision based on the initial analysis. Here the search range of w_1 would be narrowed to [0.8, 0.9] with precision at 0.01; w_2 at [0.4, 0.6] with precision at 0.025; c_1=c_2 at [1.5, 1.8] with precision at 0.01. Then conduct the 2nd grid search
6. Repeat step 1 - 4, select the optimum parameter set at the precision of 0.01 for both w_1, w_2, and c_1=c_2

Similarly, for CLPSO algorithm, the parameter is almost the same, while the refreshing gap *m* remain constant at 7. Here the inertia weight w_1, w_2 and acceleration constants c_1=c_2 were changed.

After parameter tuning, the experimental parameters of each comparison algorithm are set as shown in Table II.

TABLE II
PARAMETER SET AFTER TUNING

Algorithm	Parameter Setting
PSO	$w_0 = 0.9$, $w_1 = 0.4$, $c_1 = c_2 = 1.75$
CLPSO	$w_0 = 0.85$, $w_1 = 0.4$, $c_1 = c_2 = 1.60$, m = 7

C. Algorithm Test Results

To verify the performance using the above parameters applied to all the 10 test functions, the experiment was using the following settings:

- Dimension of function variable D = 10
- Population size N = 10
- Maximum number of iteration Max_Gen = 5000
- Test functions: $f_1 - f_{10}$

To prevent the error of the experiment, each function would run 30 times (30 repetitions) using the optimum parameter set selected in the last section. The Mean, Median, and Standard Variance of each algorithm's *gbest* value would be taken as evaluation criteria. The experiment results are shown in Table III.

The convergence images of the algorithm test based on their selected parameter set are shown in Fig. 4. The PSO and CLPSO algorithms would be compared in each test function. Here convergence plots show objective value (Y-axis in log scale) versus the number of function evaluations (X-axis linear scale).

D. Experiment Analysis

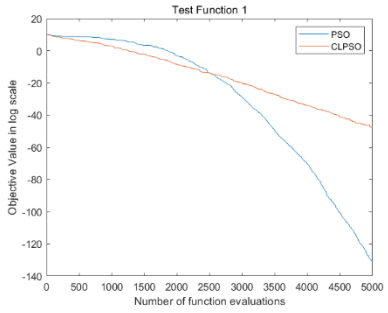
Combining the result of several evaluation criteria shown in Table III, as well as the convergence plots of the 10 test functions using the optimum parameter set derived from using grid search method in the previous sections, the global optimal search of multi-peak function like f_8 is difficult for both PSO and CLPSO algorithms. Meanwhile, if the population diversity or the total iteration number is insufficient, the PSO algorithm is easy to fall into a prematurity situation.

The result also indicated that in some test functions, the CLPSO algorithm may have worse performance than the conventional PSO algorithm such as f_1. The selection of algorithms on specific test problems is task-oriented. In this paper, the tuning progress only considered the overall impact on all 5 test functions, rather than the specific one. This may lead to the performance difference in the final result. One alternative method is to tune the parameters corresponding to each test function, and the result could be better. Therefore, we may not expect that the CLPSO has the best performance than PSO's on all classes of problems.

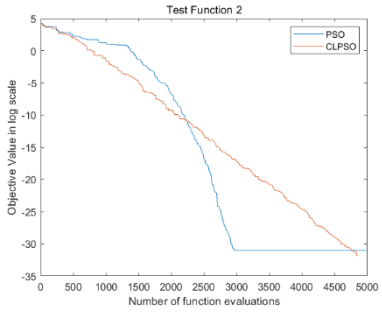
On complex test functions such as f_5 and f_6, with the updating rule, the CLPSO explores the nest search space based on the historical search experience and has better performance on these test problems.

TABLE III
RESULT FOR 10-D PROBLMES (30 REPITIONS)

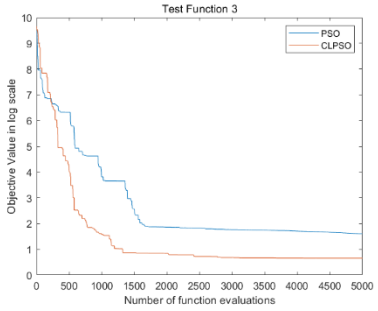
	Mean		Median		Standard Variance	
	PSO	CLPSO	PSO	CLPSO	PSO	CLPSO
f_1	2.1344e-123	9.1045e-48	7.5521e-130	1.5347e-48	1.7096e-124	1.1248e-46
f_2	5.3515e-32	4.8790e-32	0	0	1.6052e-31	7.6239e-32
f_3	7.2656e+0	3.4569e+0	9.3591e-1	2.0441e+0	3.4095e+1	9.0641e+0
f_4	4.6325e-32	9.9496e-2	4.9748e+0	0	4.1472e-1	3.0771e-1
f_5	2.4872e+2	1.5791e+1	2.3688e+2	0	1.4077e+2	4.0949e+2
f_6	1.1587e-128	5.8421e-53	3.3818e-135	3.1106e-54	1.9079e-126	1.2086e-52
f_7	2.0139e+1	1.0155e+1	2.0094e-1	1.17080e+1	1.0191e-1	8.9584e-1
f_8	3.6930e+0	2.8772e+0	3.6312e+0	2.8005e+0	2.4813e+0	4.8232e-1
f_9	9.9961e-2	1.2973e-2	8.3611e-2	4.0458e-3	5.4139e-1	1.1640e-2
f_{10}	2.4576e+1	1.7375e+1	2.6598e-1	1.0766e+0	8.8294e-2	2.8319e-2
PSO parameters: $w_0 = 0.9$, $w_1 = 0.4$, $c_1 = c_2 = 1.75$			CLSPO parameters: $w_0 = 0.85$, $w_1 = 0.4$, $c_1 = c_2 = 1.60$, m = 7			



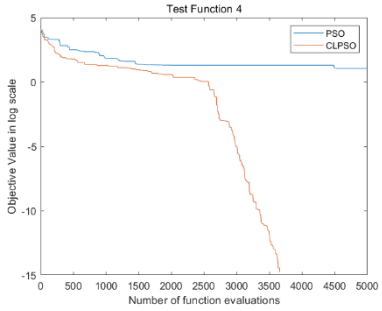
(a) f_1



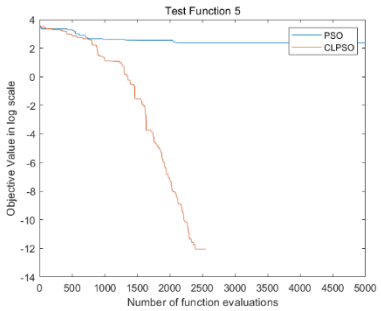
(b) f_2



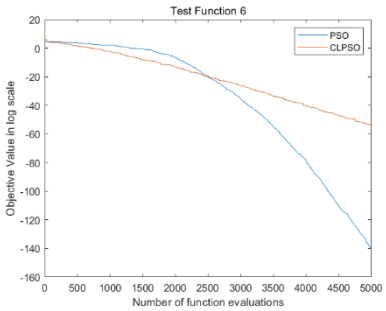
(c) f_3



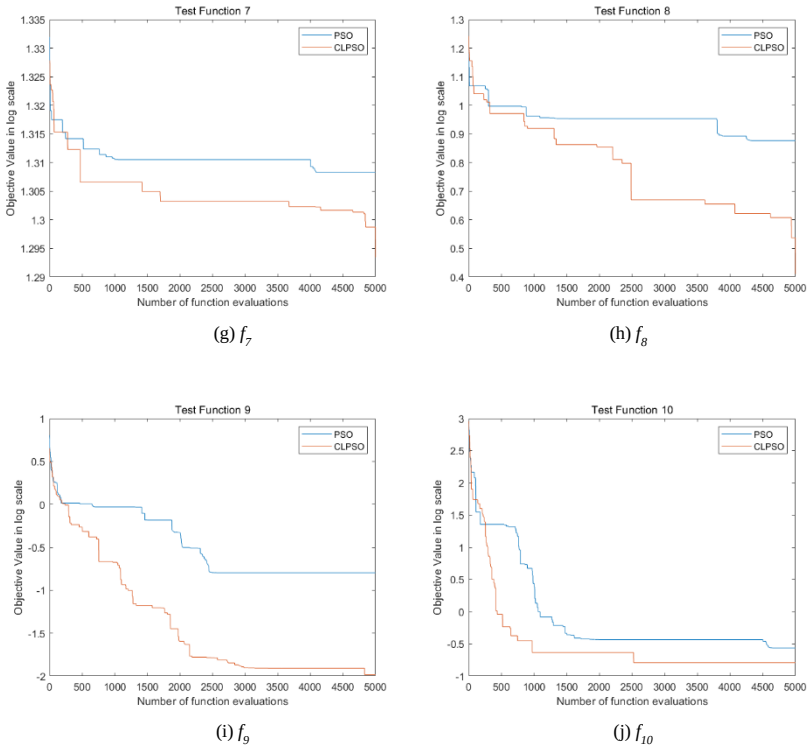
(d) f_4



(e) f_5



(f) f_6



(g) f_7 (h) f_8

(i) f_9 (j) f_{10}

Fig. 4. The mean convergence characteristic of 10-D test functions

III. Conclusion

In this paper, the comparison of PSO and CLPSO algorithms on single objective real-parameter numerical optimization was discussed. First, the fundamental parameters were selected, including the number of initial generations, number of repetitions, and maximum number of iterations. Then the 10 benchmark test problems were selected to verify the performance, among the 10 test functions, five were selected for parameter tuning. The Grid Search Method was employed to select the optimum parameter set for the minimum average error rate for the whole 5 test functions. Laster, based on the optimum parameters for PSO and CLPSO algorithms respectively, the comparison of the two algorithms' convergence performance. Finally, through the simulation and experiment of 10 test functions, the CLPSO algorithm was found to have higher efficiency than the conventional PSO algorithm most of the time. However, the selection of the algorithm is task-depended. Overall, this assignment deepened my understanding of Particle Swarm Optimization algorithms and their improved algorithms.

IV. Discussion

During the parameter tuning procedure, the grid search method was employed. However, this method lacks efficiency and would take a long time for finding the optimum parameter set. Another improved tuning method like ANOVA or iRace could be used for higher efficiency.

Besides, the criteria of tuning parameters were depended on the average error rate of all 5 test functions. More simulations and experiments can be done to optimize individual test functions' optimum parameter set.

References

[1] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In Mhs95 Sixth International Symposium on Micro Machine & Human Science, 2002.

[2] Russell C Eberhart and Yuhui Shi. Particle swarm optimization: developments, applications and resources. In Congress on Evolutionary Computation, 2002.

[3] J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions," in IEEE Trans. Evol. Comput., vol. 10, no. 3, pp.281-295, 2006.

[4] K. E. Parsopoulos and M. N. Vrahatis, "UPSO: A Unified Particle Swarm Optimization Scheme," in Lecture Series on Computational Science, pp. 868-873, 2004.

[5] T. Peram, K. Veeramachaneni, and C. K. Mohan, "Fitness-Distance-Ratio-based Particle Swarm Optimization," in Proc. Swarm Intelligence Symp., pp. 174-181, 2003.

[6] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," IEEE Trans. Evol. Comput., vol. 8, pp. 225–239, Jun. 2004.

[7] https://github.com/P-N-Suganthan/CEC2017 BoundContrained

APPENDIX I

Core Code:

Parameter Tuning: Grid Search (initial)

```
for w0= 0.5:0.1:0.1
    for w1 = 0.1:0.1:1.0
        for q = 1.4:0.1:2.0
            for funnum=1:5
            fun=funchoose(funnum);
            initial_flag=0;



            for jjj=1:10
            if shift_flag==1
                gbias=0.8.*(VRmin(fun)+(VRmax(fun)-VRmin(fun)).*rand(1,D));
%                 if fun==2
                if fun==11
                    gbias=-1+(1+1).*rand(1,D);
                end
                if fun==12
                    gbias=-500+(0+500).*rand(1,D);
                end
            else
                gbias=zeros(1,D);
            end
            cc = [q q];
%             [gbest,gbestval,fitcount] =
CLPSO_new_func(w0,w1,cc,fhd,me,Max_FES,ps,D,VRminn(fun),VRmaxn(fun),fun,VRmin(fun),VRmax(fun),gbias,norm_flag,shi
ft_flag);
%             result_temp = gbestval;
%             result=[result;result_temp];

            fun,jjj,w0,w1,cc
            % best_keep=[];best_f=1e+30;
            [CLPSO_new_gbest,CLPSO_new_gbestval,CLPSO_new_fitcount]=
CLPSO_new_func(w0,w1,cc,fhd,me,Max_FES,ps,D,VRminn(fun),VRmaxn(fun),fun,VRmin(fun),VRmax(fun),gbias,norm_flag,shi
ft_flag);
            CLPSO_new_gbestval


CLPSO_new_fitcount_res(fun,jjj)=CLPSO_new_fitcount;CLPSO_new_gbestval_res(fun,jjj)=CLPSO_new_gbestval;CLPSO_new_g
best_res(fun,jjj,:)=CLPSO_new_gbest;

            end
            end

            mean(CLPSO_new_gbestval_res')
            result_temp = [result_temp;w0];
            result_temp = [result_temp;w1];
            result_temp = [result_temp;q];
            result= [result;result_temp'];
            result_temp = [];
            result = [result,mean(CLPSO_new_gbestval_res')]
            result_total=[result_total;result];
            result = [];


        end
    end
end
```