8th International Conference on Advanced Computational Intelligence
Chiang Mai, Thailand; February 14-16, 2016

A Dynamic Competitive Swarm Optimizer Based-on Entropy for Large Scale Optimization

Wen-Xiao Zhang
School of Data and Computer Science
Sun Yat-sen University
Guangzhou, China

Wei-Neng Chen* and Jun Zhang
Key Lab. of Machine Intelligence and Advanced Computing, Ministry of Education
Guangzhou, China
Email: cwnraul634@aliyun.com

***Abstract*—In this paper, a dynamic competitive swarm optimizer (DCSO) based on population entropy is proposed. The new algorithm is derived from the competitive swarm optimizer (CSO). The new algorithm uses population entropy to make a quantitative description about the diversity of population, and to divide the population into two sub-groups dynamically. During the early stage of the execution process, to speed up convergence of the algorithm, the sub-group with better fitness will have a small size, and worse large sub-group will learn from small one. During the late stage of the execution process, to keep the diversity of the population, the sub-group with better fitness will have a large size, and small worse sub-group will learn from large group. The proposed DCSO is evaluated on CEC'08 benchmark functions on large scale global optimization. The simulation results of the example indicate that the new algorithm has better and faster convergence speed than CSO.**

Keywords—competitive swarm optimizer; population entropy ; sub-group; pairwise competition; large scale optimization

I. INTRODUCTION

Particle swarm optimizer (PSO) is an evolutionary algorithm, introduced by Kennedy and Eberhart in [1] and [2]. The algorithm derives from the behavior of social animals like bird flocking and fish schooling. The algorithm contains a swarm of particles, which are randomly initialized in an n-dimensional search space. Each particle has a velocity vector and a position vector. During the each generation, each particle update its velocity and position by learning from the particle's own best position and the historically best solution of the whole swarm. The two vectors of each particle are updated using the following equations:

$$V_i(t+1) = \omega \cdot V_i(t) + c_1 \cdot r_1\left(pBest_i - X_i(t)\right) + c_2 \cdot r_2 \cdot (gBest - X_i(t)) \quad (1)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (2)$$

where $pBest_i$ is the i-th particle own historically best position, $gBest$ is the historically best solution of the whole swarm. c_1 and c_2 are two parameters called learning factors, which keep a delicate balance between $pBest_i$ and $gBest$. r_1 and r_2 are random numbers uniformly distributed in [0,1]. w is a parameter called the inertia weight.

Due to its simplicity in program implementation, PSO has become one of the most powerful optimization algorithms and has been widely used in many subjects and engineering fields [3]. However, many optimization problems in real world applications are of large scale [4]. It has been found that PSO's performance becomes poor when the number of decision variables grows [5]. In many cases, the growing of dimensionality in an optimization problem will significantly increase the number of local optima, resulting in premature convergence of PSO [5].

A natural approach to solve high-dimensional optimization problems is to adopt a divide-and-conquer strategy. One famous attempt is the cooperative co-evolution (CC) approach [6]. It decomposes a large scale problem into smaller sub-problems and solves these sub-problems separately. CC approach is efficient when the variables are independent from each other, but becomes less efficient if the problem is a non-separable. This is because in non-separable problem it is not easy to let non-separated variables optimized in just the same groups. To improve the performance of CC approach, it is necessary to make interacting variables in the same group. For the original random grouping strategy for CC approach, Omidvar [7][8] drew the conclusion that the probability of grouping interacting variables in one sub-group using random grouping dropped significantly as the number of interacting variables increased. Later, several attempts have been made to provide better grouping schemes for the CC approach, e.g., differential grouping [9], variable interaction learning mechanism [10]. These approaches use a decomposition phase to estimate the relationship among variables and group accordingly. They have shown promising performance on some benchmark problems. But in the problems with more complicated fitness landscape, (e.g., the correlations of variables change in different areas of the search space) it is difficult for such approaches to get accurate groups. In a word, the CC approach provides an efficient divide-and-conquer way for large-scale optimization, but it remains a big challenge to

*Corresponding Author.
This work was supported in part by the NSFC projects Nos. 61379061, 61332002, 61511130078, in part by Natural Science Foundation of Gungdong for Distinguished Young Scholars No. 2015A030306024, in part by the "Guangdong Special Support Program" No. 2014TQ01X550, and in part by the Guangzhou Pearl River New Star of Science and Technology No. 201506010002.

Authorized licensed use limited to: Hanyang University. Downloaded on November 21,2023 at 06:40:17 UTC from IEEE Xplore. Restrictions apply.

design effective grouping techniques for problems with complicated fitness landscape.

Apart from the CC approach, designing an effective learning strategy for large-scale optimization is also very important, as there are much more local optima in large-scale problems and thus an efficient learning strategy should have the ability to improve search diversity. One representative approach is the competitive swarm optimizer (CSO) developed in [11]. It doesn't update particles' velocity with $pBest_i$ and *gBest*. This is because in the large scale optimization the particles following the *gBest* would converge too fast to get out of local optima. In view of this, it introduces a pairwise competition mechanism. In each generation, the particles would be divided into couples randomly. Afterwards, a competition is made between the two particles in each couple. After the competition, the particle with better fitness is known as winner, the other is called loser. Winner would get to the next generation directly, however, loser would update its velocity vector and position vector. CSO keeps the simplicity in program implementation and have better performance than CC approach in non-separable functions. However, CSO faces the problem of slow convergence.

From the above discussions, it can be seen that the CSO provides a new and promising learning strategy for avoiding premature convergence in large-scale optimization. But it remains room for further improving in its convergence speed. One important reason for the slow convergence of CSO is that in each generation only half of particles which lose in competition can be updated, and the rest winning particles remain unchanged. If the information of such winning particles can be exploited, it is possible that the search speed of CSO can be improved.

Following this idea, this paper intends to propose a new algorithm called dynamic competitive swarm optimizer (DCSO). It uses population entropy to divide the population into two sub-groups dynamically. The sub-group with worse fitness will learn from better sub-group, and better sub-group will learn on its own. During the early stage of the execution process, as population entropy is high, the sub-group with better fitness will have a small size, and large worse sub-group will learn from small one.In this way, the new algorithm is similar to PSO. Then, during the late stage of the execution process, as population entropy is low, the sub-group with better fitness will have a big size, and the other small worse sub-group will learn from big one. The algorithm turns into total CSO.

The rest of this paper is organized as follows: Section II presents the basic CSO algorithm and definition of population entropy. Section III develops the DCSO algorithm in detail. Section IV experimentally validates the DCSO and compares it with CSO on 7 CEC'08 benchmark functions of dimension of up to 1000.Conclusions are drawn in Section V.

II. PRELIMINARY

A. Basic Competitive Swarm Optimizer

Cheng and Jin[11] proposed the concept of pairwise competition mechanism and introduced a approach: To solve the problem, a swarm $P(t)$ contains m particles is randomly initialized, where m is its swarm size and t is generation index. In each generation the particles are randomly allocated into $m/2$ couples (assuming that the swarm size m is even number). Afterwards, a competition is made in each couple. The particle having a better fitness called winner will pass to the next generation directly. And the loser in each couple will learn from the winner use following equations:

$$V_l(t+1) = r_1 \cdot V_l(\mathrm{t}) + r_2 \cdot \left(X_w(t) - X_l(t)\right) + \varphi \cdot r_3 \cdot (\bar{X}(t) - X_l(t)) \quad (3)$$

$$X_l(t+1) = X_l(t) + V_l(t+1) \quad (4)$$

where $V_l(t)$ and $X_l(t)$ represent velocity and position vectors of the loser, and $V_w(t)$ and $X_w(t)$ represent velocity and position vectors of the winner. r_1, r_2, r_3 are random numbers uniformly distributed in [0,1]. φ is a parameter that controls the influence of $X(t)$. $X(t)$ is mean position value of relevant particles. It has two versions: a global version and a local version. However, experiments show that the diversity of the global CSO is already sufficient and additional diversity may slow down the convergence. Therefore, $X(t)$ usually denotes as the global mean position of all particles.

There are two parameters in CSO, namely, the swarm size m and the social factor φ. If the swarm size is too small, the particles tend to converge very fast, thus leading to premature convergence. On the contrary, if the swarm size is too large, a large number of fitness evaluations(FEs) will be required during each generation, which may be impractical and waste time. The values of their combination that were proposed in [11] is summarized in Table I. From the result, it can be seen that non-separable functions (*f2, f3,* and *f7*) require a smaller φ than separable functions (*f1, f4, f5,* and f6) .The reason might be that separable functions are easier to optimize, as a result, a bigger φ would work better because it leads to faster convergence.

TABLE I. PARAMETER SETTINGS OF CSO FOR SEVEN FUNCTIONS OF 100-D,500-D,1000-D

Parameters	Dimensions	Separable functions	Non-separable functions
m	100-D 500-D 1000-D	100 250 500	100 250 500
φ	100-D 500-D 1000-D	0 0.1 0.15	0 0.05 0.1

B. The Definition of Population Entropy

Entropy is a measure of system complexity. Many researches have introduced entropy in PSO. For example, Ran and Wang[12] used population entropy to describe the location confusion degree of particles, and it could reduce the waste of search space of PSO. Entropy was also applied in [13] to measure the diversity of the whole population and guide the particles to migrate.

Authorized licensed use limited to: Hanyang University. Downloaded on November 21,2023 at 06:40:17 UTC from IEEE Xplore. Restrictions apply.

The definition of population entropy is as follow:

In the set R which consists of the fitness of all particles exists subsets A_1, A_2 ... A_n, satisfied the following equations:

$$A_1 \bigcup A_2 \bigcup \ldots \bigcup A_n = R,\ A_1 \bigcap A_2 \ldots \bigcap An = \phi$$

In the subsets A_1, A_2 ... A_n, there are corresponding elements s_1, s_2 ... s_n.

The definition of the population entropy is:

$$E = -\sum_{i=1}^{n} p_i \log(p_i) \tag{5}$$

$$p_i = s_i / \sum_{i=1}^{n} s_i \tag{6}$$

It can be found that the population entropy is 0 when the fitness values of all particles are equal. The higher population entropy means particle swarm have more diversity and stronger exploration. In order to make population entropy normalized, we can use the following equation:

$$E = -\sum_{i=1}^{n} p_i \log_n(p_i) \tag{7}$$

In (7), the maximum of E is 1, which can be achieved when $p_1 = p_2 \ldots = p_n = 1/n$.

III. Algorithm

The new algorithm proposed in this paper differs from the CSO in that more than a half of particles are updated in one generation. The sizes of two sub-groups are adaptively adjusted according to the population entropy. When the population entropy is high, worse particles only need to learn from several particles. When the population entropy is small, it means the diversity of swarm is bad. Therefore, the worse particles should choose one from a large range of particles. The main idea is showed in Fig.1.

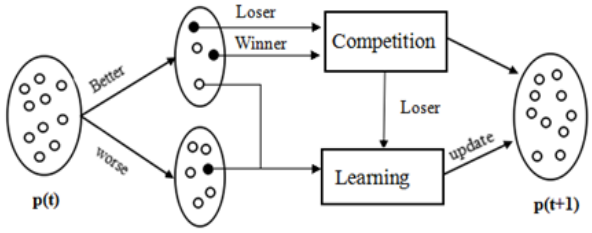


Fig. 1.The main idea of DCSO, In each generation particles in better sub-group would compete in couples. Winner would pass to p(t+1) directly and loser would learn from winner. For particles in worse sub-group, each will learn from particles in better group.

For convenience, we denotes $p_b(t)$, $p_w(t)$ as better sub-group and worse sub-group. The swarm size of $pb(t)$ is m_b, and the size of $pw(t)$ is m_w. In each generation, the size of $pb(t)$ and $pw(t)$ will be calculated using the following strategy:

$$m_b = \begin{cases} -(m/d) * PE(t) + m/d & PE(t) > 1-d \\ m & PE(t) \le 1-d \end{cases} \tag{8}$$

$$m_w = m - m_b \tag{9}$$

where $PE(t)$ is the normalized population entropy of generation t. And d is a factor that controls the increasing speed of m_b, usually less than 1. It is easy to see that the minimum value of m_b is 0, if and only if $PE(t)$ equals 1. Then, each particle in worse sub-group will update the velocity and position by randomly choosing a particle in better sub-group to learn from. The update rule is shown (10) and (11).

$$V_i(t+1) = r_1 \cdot V_i(t) + r_2 \cdot (X_j(t) - X_i(t)) + \varphi \cdot r_3(\overline{X}(t) - X_i(t)) \tag{10}$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{11}$$

where $Xj(t)$ is the randomly choosing particle in better sub-group $pb(t)$. The update strategy is similar to PSO, where $Xj(t)$ is just like $pBest$ and $X(t)$ is just like $gBest$. For particles in better sub-group $pb(t)$, it will adopt pairwise competition mechanism. In each couple, the winner will go through to the next generation directly, and the loser will update using following learning strategy:

$$V_l(t+1) = r_1 \cdot V_l(t) + r_2 \cdot (X_w(t) - X_l(t)) \tag{12}$$

$$X_l(t+1) = X_l(t) + V_l(t+1) \tag{13}$$

The pseudo-code is showed in Fig.2 and steps involved are given as follows:

Step 1 Initialization: The initial positions of all particles are generated randomly within the n-dimensional search space, with velocity initialized to 0.

Step 2 Calculation: Calculate each particle fitness and the population entropy of swarm.

Step 3 Swarm division: Sort particles by fitness value. Then use rule (8) and (9) to adjust the size of two sub-groups.

Step 4 Worse sub-group updating: For each particles in worse sub-group follows the velocity update rule (10) and the position update rule (11) to adjust its velocity and position .

Step 5 Better sub-group updating: For each particles in better sub-group follows the velocity update rule (12) and the position update rule (13) to adjust its velocity and position .

Step 6 Terminal Condition Check: If the number of FEs is large than the predefined maximum number, the algorithm terminates. Otherwise, go to Step 2 for a new generation.

IV. Experiment Studies

To test the performance of proposed algorithm, we do a set of experiments on seven benchmark functions proposed in CEC'08 special session on large scale optimization problems [14]. Among the 7 functions, we can divide them into 2 groups. *f1*, *f4* and *f6* are in first group, which are separable functions. Other 4 functions are in another group, which are non-separable functions.

All experiments are implemented on a PC with an Intel Core i5-3210M 2.5GHZ CPU, 6GB RAM and Microsoft

Authorized licensed use limited to: Hanyang University. Downloaded on November 21,2023 at 06:40:17 UTC from IEEE Xplore. Restrictions apply.

Windows 7 Home SP1 64-bit operating system, and DCSO is written in language C on Visual Studio 2010.

All results are averaged 25 independent runs. During each run, the maximum number of fitness evaluations (FEs), as recommended by [13], is set to 5000*D, where D is the search dimension of functions.

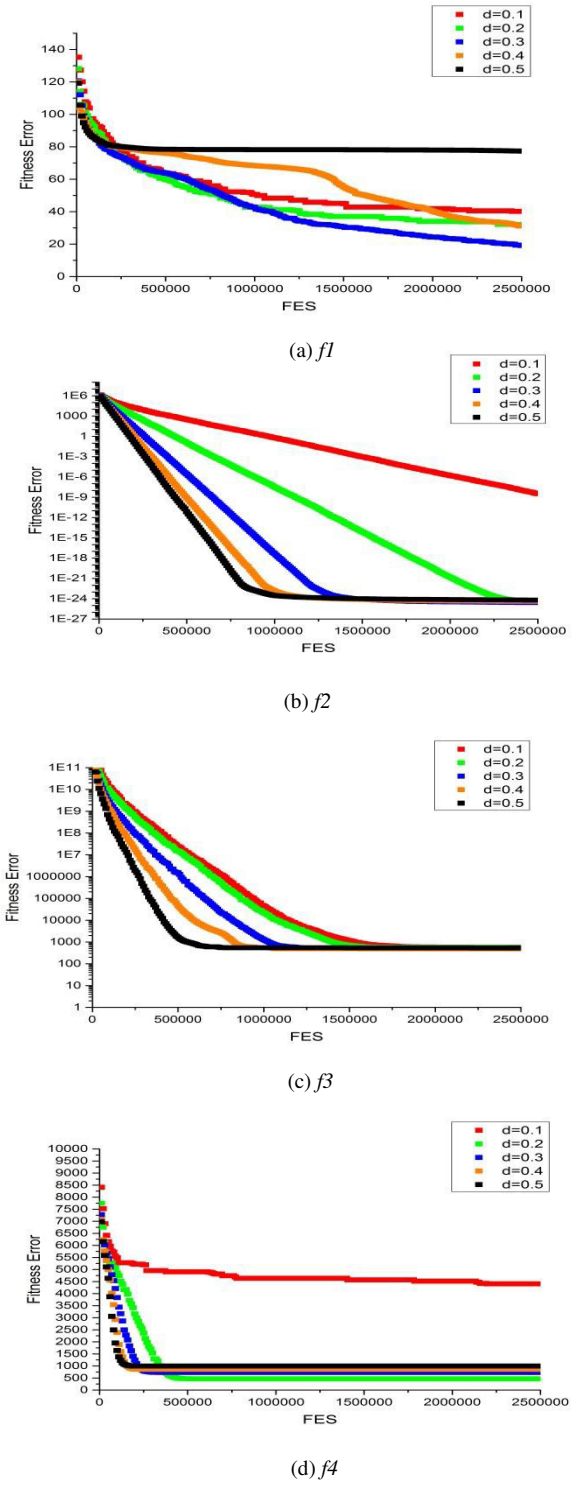
```
procedure generate
1.  t=0;
2.  randomly initialize p(0);
3.  while terminal condition is not satisfied do
4.        calculate the fitness of all particles in p(t);
5.        sort particles in p(t);
6.        calculate the population entropy of p(t) using (7);
7.        divide p(t) into pb(t) and pw(t) using (8),(9);
8.        for each particle Xi(t) in pw(t) do
9.            randomly choose one particle Xj(t) from pb(t);
10.           update Vi(t) and Xi(t) using (10) and (11);
11.           add Xi(t) into p(t+1);
12.       end for
13.       while pb(t) is not none do
14.          randomly choose two particles X1(t) and X2(t) ;
15.          if f(X1(t)) < f(X2(t)) then
16.             Xw(t) =X1(t), Xl(t) =X2(t);
17.          else
18.             Xw(t) =X2(t), Xl(t) =X1(t);
19.          end if
20.          update Vl(t) and Xl(t) using (12)and (13);
21.          add Xw(t) and Xl(t) into p(t+1);
23.          remove X1(t) and X2(t) from pb(t);
24.       end while
25.       t=t+1;
26.end while
end procedure
```

Fig. 2. The pseudo code of the Dynamic Competitive Swarm Optimizer

A. Parameter Settings

From (8), it is obvious that the factor d has an effect on the size of two sub-groups. With a large d, the size of better sub-group increases slow, and the particles tend to converge very fast before the search space is well explored, thus leading to premature convergence. On the contrary, if the factor d is too small, the size of better sub-group increases fast, which will make convergence speed as slow as CSO. In order to investigate the influence of factor d, we test the CEC'08 functions with dimensions of 500 with d varying from 0.1 to 0.5. Since DCSO is derived from CSO, the value of swarm size m and social factor φ in DCSO is similar to CSO. Therefore, the parameter setting in Table 1 is adopted.

In Fig.3, the statistic results of the fitness error obtained by DCSO with different value of factor d are studied. According to the results, it seems that *f1*, *f3* require a small *d,* The reason might be that *f1* and *f3* are easier to optimize, as a result, smaller d would work better because it leads to faster convergence.



(a) *f1*

(b) *f2*

(c) *f3*

(d) *f4*

Fig. 3. Fitness error on 4 functions *f*1, *f2, f3* and *f4* of 500-D with factor d varying from 0.1 to 0.5

Authorized licensed use limited to: Hanyang University. Downloaded on November 21,2023 at 06:40:17 UTC from IEEE Xplore. Restrictions apply.

In order to better understand the relation between population entropy *PE(t)* and factor *d*. We investigate the variation of population entropy in CSO. The result is shown in Fig 4. Since CSO doesn't use *pBest* and *gBest* to update particles and randomly choose couples to compete, it will definitely make CSO have high population entropy, which means a high diversity. That's why CSO has powerful performance in large scale optimization, and it is common that the population entropy is over 0.8 when the convergence speed is fast. More interesting, there is a sharply decrease in f5 when it gets into premature convergence.

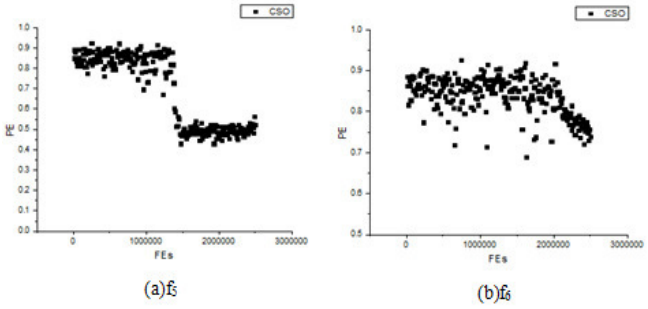


Fig. 4. The population entropy of non-separable function *f5* and separable function *f6* of 500-D using CSO

So in DCSO we adopt a strategy: If population entropy is lower than 0.65, the size of better sub-group sets as whole swarm. This is because it means the swarm tends to fall into local optima. And if population entropy is close to 1, the size of better sub-group will just have a few particles. Hence, in (8), it go through (1,0) and (0.65,*m*).Using linear interpolation, it can easily get the slope of the line is:

$$K = -m/0.35$$

Thus, the factor *d* is set to 0.35 if dimension is 500. The population entropy of DCSO with d equals is showed in Fig.5. Furthermore, the best combinations in DCSO are summarized in Table II.

TABLE II. PARAMETER SETTINGS OF DCSO FOR SEVEN FUNCTIONS OF 100-D,500-D,1000-D

Parameter	Dimensions	Separable functions	Non-separable functions
m	100-D 500-D 1000-D	100 250 500	100 250 500
φ	100-D 500-D 1000-D	100 250 500	100 250 500
d	100-D 500-D 1000-D	0.25 0.35 0.45	

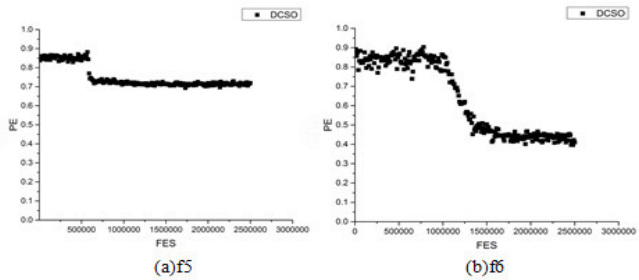


Fig. 5. The population entropy of non-separable function *f5* and separable function *f6* of 500-D using DCSO

B. Comparison Results

In order to verify the performance of DCSO for large scale optimization, DCSO have compared with CSO on CEC'08 test functions with dimensions of 100, 500 and 1000. Based on the previous experiments, we set the factor *d* with 0.25, 0.35, and 0.45 for 100-D, 500-D, 1000-D. The fitness error is summarized on Table III, Table IV, Table V.

The results of fitness error show that DCSO have better performance in comparison with CSO on *f1*, *f2*, *f3* and *f6*. Especially, DCSO has significantly better performance on f1 and f6. Moreover, DCSO and CSO have similar performance on f5. However, regardless of the number of the dimensions, DCSO has poor performance on f4 in comparison with CSO, which is a shifted Rastrigin function with a large number of local optima. The probable reason is that pair competition mechanism has poor performance on f4. Although the diversity of CSO is strong, it still converge too fast to jump out of local optima with a problem has a huge number of Optima. Thus, speeding up the convergence will aggravate premature convergence.

In additional, the convergence profiles of two typical separable functions (f1, f6) and two non-separable functions (f2, f5) are showed in Fig.6. It can be seen that, the convergence speed of DCSO is much faster than CSO and have better performance than CSO.

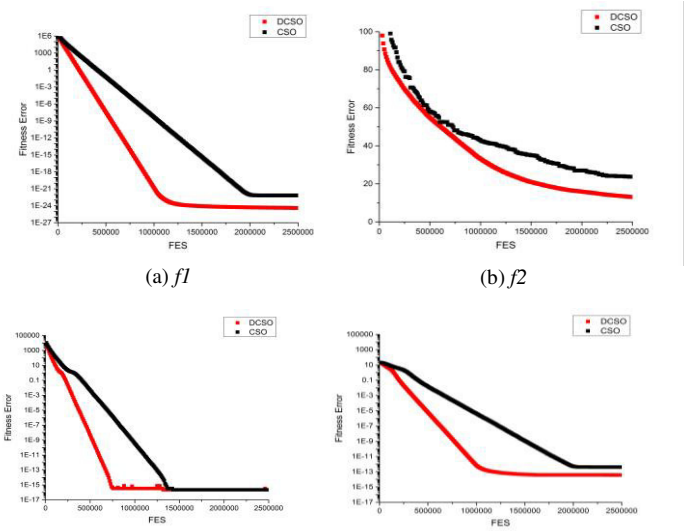


Fig. 6. The convergence profits of DCSO and CSO on 500-D Functions *f1, f2, f5, f6*

Authorized licensed use limited to: Hanyang University. Downloaded on November 21,2023 at 06:40:17 UTC from IEEE Xplore. Restrictions apply.

TABLE III. THE STATISTICAL RESULTS (MEAN VALUES, BESR VALUES AND STANDARD DEVIATIONS) OF FITNESS ERROR ON 100-D TEST FUNCTIONS

100-D		f1	f2	f3	f4	f5	f6	f7
DCSO	Mean	**0.00E+00**	**2.57E+01**	**3.19E+02**	1.12E+02	0.00E+00	**1.10E-14**	-7.77E+05
	Best	0.00E+00	2.04E+01	8.47E+01	9.65E+01	0.00E+00	7.55E-15	-
	Std	0.00E+00	1.97E+00	3.33E+02	1.37E+01	0.00E+00	6.96E-16	1.34E+04
CSO	Mean	1.72E-28	3.39E+01	3.46E+02	**5.19E+01**	0.00E+00	1.20E-14	-7.46E+05
	Best	0.00E+00	2.33E+01	8.96E+01	3.48E+01	0.00E+00	1.11E-14	-
	Std	1.73E-28	5.88E+00	4.85E+02	6.25E+00	0.00E+00	1.52E-15	1.67E+04

TABLE IV. THE STATISTICAL RESSULTS (MEAN VALUES, BEST VALUES AND STANDARD DEVIATIONS) OF FITNESS ERROR ON 500-D TEST FUNCTIONS

500-D		f1	f2	f3	f4	f5	f6	f7
DCSO	Mean	**3.54E-25**	**1.98E+01**	**5.26E+02**	7.37E+02	2.22E-16	**3.86E-14**	-2.07E+06
	Best	2.84E-25	1.32E+01	4.84E+02	6.87E+02	2.22E-16	3.24E-14	-
	Std	5.54E-26	6.23E+00	2.70E+01	3.72E+01	0.00E+00	2.36E-15	2.81E+04
CSO	Mean	6.55E-23	2.55E+01	5.67E+02	**3.24E+02**	2.22E-16	4.13E-13	-1.97E+06
	Best	6.13E-23	2.22E+01	4.89E+02	2.86E+02	2.22E-16	3.98E-13	-
	Std	2.93E-24	4.08E+00	4.73E+01	1.91E+01	0.00E+00	8.17E-15	4.08E+04

TABLE V. THE STATISTICAL RESSULTS (MEAN VALUES, BEST VALUES AND STANDARD DEVIATIONS) OF FITNESS ERROR ON 1000-D TEST FUNCTIONS

1000-D		f1	f2	f3	f4	f5	f6	f7
DCSO	Mean	**1.83E-24**	**3.37E+01**	**9.80E+02**	1.53E+03	5.55E-16	**8.57E-14**	-2.07E+06
	Best	1.74E-24	3.32E+01	9.74E+02	1.49E+03	5.53E-16	8.24E-14	-
	Std	2.62E-25	4.06E-01	3.65E+01	6.57E+01	5.23E-17	2.36E-15	2.81E+04
CSO	Mean	1.09E-21	4.07E+01	9.90E+02	**6.96E+02**	**2.26E-16**	1.22E-12	-3.86E+06
	Best	1.04E-21	4.01E+01	9.86E+02	6.73E+02	2.22E-16	1.20E-12	-
	Std	3.77E-23	4.72E-01	2.73E+01	2.47E+01	2.19E-17	2.27E-14	4.78E+04

Authorized licensed use limited to: Hanyang University. Downloaded on November 21,2023 at 06:40:17 UTC from IEEE Xplore. Restrictions apply.

V. CONCLUSION

In this paper, we proposed a dynamical algorithm DCSO based on population entropy. It manages to prevent premature convergence and keep the fast convergence. The algorithm adopts pairwise competition mechanism, and uses population entropy to divide swarm into better sub-group and worse sub-group dynamically. In order to confirm the effectiveness and performance of DCSO, we compared it with CSO based on 7 CEC 08's benchmark functions. Experiment results show that the new algorithm can have faster convergence speed and the performance of DCSO is much better than CSO on some function.

REFERENCES

[1] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in Proceedings of International Symposium on Micro Machine and Human Science. IEEE, 1995, pp. 39–43.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in Proceedings of the IEEE International Conference on Neural Networks, vol. 4. IEEE, 1995, pp.1942-1948.

[3] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources" in Proceedings of the IEEE Congress on Evolutionary Computation. IEEE Service Center, 2001, pp.81~86.

[4] E. Sayed, D. Essam, and R. Sarker., "Dependency Identification technique for large scale optimization problems," in Proceedings of the 2012 IEEE Congress on Evolutionary Computation(CEC), IEEE, Brisbane, 2012.

[5] F. Van den Bergh and A. P. Engelbrecht, "A Cooperative approach to particle swarm optimization," IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 225 "239, 2004.

[6] W.-N. Chen, et al., "Particle swarm optimization with an aging leader and challengers," IEEE Transactions on Evolutionary Computation, vol. 17, no. 2, pp. 241–258, 2013.

[7] M. A. Potter and K. A. De Jong, " A cooperative coevolutionary approach to function optimization," Parallel Problem Solving from Nature, PPSN III, pp. 249–257, 1994.

[8] M. N. Omidvar, X. D. Liu, X. Yao, and Z. Y. Yang, "Cooperative Coevolution for large scale optimization through more frequent random grouping," in Proceedings of 2010 IEEE Congress on Evolutionary Computation(CEC), IEEE, 2010, pp. 1 "8.

[9] M. N. Omidvar, X. Li, Y. Mei, et al., "Cooperative co-evolution with differential grouping for large scale optimization". Evolutionary Computation, IEEE Transactions on, 18(3): 378-393, 2014.

[10] W. Chen, T. Weise, Z. Yang, et al. "Large-scale global optimization using cooperative coevolution with variable interaction learning", Parallel Problem Solving from Nature, PPSN XI. Springer Berlin Heidelberg, 2010, pp. 300-309.

[11] R. Cheng and Y. Jin, "A Competitive Swarm Optimizer for Large Scale Optimization," IEEE Transactions on Cybernetics , vol. 45, no.2, pp. 191-204, 2014.

[12] M. Ran, Q. Wang, and C. Dong, "A dynamic search space Particle Swarm Optimization algorithm based on population entropy," in Proceedings of the 26th Chinese Control and Decision Conference, 2014, pp.4292-4296

[13] C. Hu, M. Zhao, and Y. Wang, "Co-evolutionary particle swarm optimization based on population entropy" in Proceedings of the 26th Chinese Control Conference, 2008, pp. 70-74

[14] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang, "Benchmark functions for the cec'2008 special session and competition on large scale global optimization," Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.

Authorized licensed use limited to: Hanyang University. Downloaded on November 21,2023 at 06:40:17 UTC from IEEE Xplore. Restrictions apply.