Particle Swarm Optimization with Multiple Adaptive Sub-swarms

Xunlin Jiang*, Yingjie Yue, Yinxing Min, Qingyu Zhang, Xinxing Gui

Army Infantry College of PLA, No.2001 Changzheng West Road, Nanchang, 330100, China

*xunlinjiang@163.com

Abstract. Benefiting from its simplicity and efficiency, particle swarm optimization (PSO) algorithm has shown great performance on various problems. However, for different optimization problems or different search areas, it is still difficult to achieve a satisfying trade-off between exploration and exploitation. On the basis of canonical PSO algorithm, a variety of improved algorithms have been proposed, which have different capabilities of exploitation and exploration, and each of them performs effective in some problems. This paper proposes a particle swarm optimization with multiple adaptive sub-swarms (PSOMAS). It uses multiple sub-swarms strategy, in which each sub-swarm is evoluted by different algorithms, and an adaptive strategy is also used to reduce the consumption of computing resources. A comprehensive experimental study is conducted on 30 benchmark functions, to compare with several well-known variants of PSO algorithms. The results show that PSOMAS with RT=100 could obtain a better overall performance than all others. Moreover, PSOMAS could find high-quality solution in different problems by varying the value of RT.

Keywords: Particle swarm optimization (PSO), Multiple sub-swarms, Adaptive strategy, Dynamic update mechanism.

1. Introduction

Since particle swarm optimization (PSO) algorithm was proposed in 1995[1], it has become an efficient method for searching approximate optimal solution due to its simplicity and flexibility. PSO algorithm achieves a more satisfactory result in parameter optimization problems of high dimensional space, which benefits from its simplicity and efficiency. Some survey show that PSO is widely used in various fields, such as neural network training[2], cloud computing[3], fault diagnosis[4], data mining[5], big data application[6] etc.

After years of development, PSO algorithm is improved from the aspects of inertia weight, learning factor and network topology structure, and achieves an excellent state[7]. The improvement of single algorithm is difficult to achieve breakthroughs. As a result, researchers have begun to hybrid different algorithms to enhance the optimization ability[8].PSO with multiple sub-swarms shows better diversity, which is faster and more efficient in solving multimodal optimization problems[9]. However, according to “no free lunch” (NFL) theorems, if an algorithm is improved in some aspect which makes it more suitable to solve certain problems, its ability to solve other problems is bound to be weakened[10]. Multiple sub-swarms hybrid algorithm enhances the global optimization ability. However, it
also brings the time consumption superposition which is produced by the simultaneous optimization of different sub-swarms. To reduce the time consumption, the sub-swarms which have poor performance will gradually withdraw from the operation in the iterative process, so as to increase the operation of those who have better performance[11]. Once an PSO algorithm has been proposed, the capabilities of exploration and exploitation has been set down, and the performance is not the same while optimizing different problems. In a certain problem, one algorithm may show better exploration capability and another algorithm may show better exploitation capability. So if there is an algorithm which hybrid of the two different algorithms, it could gain the capabilities of the different algorithms, so that the complicated optimization problem could be solved.

This paper proposes a novel PSO algorithm that uses multiple sub-swarms adaptive strategy, in which each sub-swarm is trained by different algorithms, so as to make full use of the different capabilities of the different algorithms. In the training, different sub-swarms can achieve information sharing through the position of global optimum, an adaptive hybrid strategy is used to select the best algorithms in the iterative process, which ultimately reduces the overall time consumption.

2. Related works

When PSO algorithm is used to optimize the practical problems, any feasible solution is formed as a particle, and it achieves the best solution by moving the particles’ positions. Through local search and global search, the algorithm shows the capability of exploration and exploitation. In particle swarm, each particle contains two properties in the search space, which are location $x_i$ and speed $v_i$ respectively, and the particle adjusts its speed first during the iteration as follows:

$$v_i(n + 1) = v_i(n) + c_1 r_1(p_i - x_i) + c_2 r_2(g - x_i)$$

(1)

Where $n$ represents the iteration step number, $c_1$ and $c_2$ are local and global learning factor respectively, $r_1$ and $r_2$ are random numbers between $[0, 1]$, $p_i$ stands as the previous best value, and $g$ stands as the global best value. The particle’s position is updated as follows:

$$x_i(n + 1) = x_i(n) + v_i(n)$$

(2)

During the iterative process, the particles would fly over the search range, and the particle velocity would become extremely large or small, which would lead to the decreasing of search efficiency or even the loss of the capability to search the global best. On the basis of canonical PSO algorithm, a variety of improved algorithms have been proposed. In the remainder of this section we briefly introduce some well-known ones.

In the traditional particle swarm, a particle with $k$ neighbors selects one particle as the source of the effect, ignoring the other particles. In this case, the more neighbors there are, the one picked is more likely to be the better one. In the full-informed particle swarm optimization (FIPSO) algorithm, however, all neighbors are a source of influence[12]. For particle $i$, the size of its neighbors $L_i$ is $N_i$, whenever it finds a good area in search space, it will only directly affect its neighbors $L_i$, but not all other particles. The velocity of the particle $i$ is updated as follows

$$v_i(n + 1) = \chi(v_i(n) + \frac{1}{N_i} \sum_{j \in L_i} \varphi_{r_j}(p_j - x_i))$$

(3)

That means it doesn’t need to share the velocity adjustments between two terms, which contain the personal best position and the global best position.
Most stochastic optimization algorithms like genetic algorithm and differential evolution algorithm suffer from the “curse of dimensionality”, which simply means that their performance deteriorates as the dimensionality of the search space increases. Cooperative particle swarm optimization (CPSO) algorithm recommends partitioning the search space by splitting the solution vector into smaller vectors [13]. In CPSO, the particles interact by transmitting information to each other while solving problems, even the information exchanged between particles may be incorrect. For each particle, the personal best position $p_i$ is updated as follows:

$$p_i = \begin{cases} 
\frac{b(p_i, x_i, K)}{f(b(g, x_i, K))} & \text{if } f(b(p_i, x_i, K)) < f(b(g, x_i, K)) \\
p_i & \text{else}
\end{cases}$$

(4)

Where $b(p_i, x_i, K)$ is the interact function, and $K$ represents the dimensions that are corresponding to the communication information.

Comprehensive Learning Particle Swarm Optimization (CLPSO) is first introduced by Liang[14], which is a state-of-the-art metaheuristic that encourages a particle to learn from different exemplars on different dimensions[15]. CLPSO computes the fitness value of a particle and updates the particle’s personal best position only when it’s feasible. Because all examples are feasible, particles will eventually be pulled back into the search space when they are not feasible. Whether the particle learn by itself or from other particles depends on a learning probability $P_C$. Like FIPSO, CLPSO regards that if all particles learn from the global best position, the swarm is likely to fall into local optimum traps. So in CLPSO, the particle velocity only learn from the personal best position.

$$v_i(n + 1) = w \cdot v_i(n) + c \cdot r \cdot (p_i - x_i)$$

(5)

Where $w$ is the inertia weight, $c$ is the learning factor, and $r$ is a vector constituted by random number between $[0, 1]$. CLPSO is able to locate the global optimum region for many complex multimodal problems as it is excellent in preserving the particles’ diversity and thus preventing premature convergence.

In PSO, global searches are performed with larger inertial weights, while local searches are performed with smaller inertia weights. A adaptive inertial weight particle swarm optimization (AIWPSO) is proposed which uses the success rate of the swarm as its feedback parameter to ascertain the particles’ situation in the search space [16].

$$w = w_{\text{max}} + (w_{\text{min}} - w_{\text{max}}) S_n / N$$

(6)

Where $S_n$ represents the number of particles which successfully updated their position in the $n^{\text{th}}$ iteration, and $N$ is the total count of the swarm. The empirical studies on benchmark problems and real engineering problems show that AIWPSO can adapt the value of $w$ to dynamic and static environments quite effectively.

The particle swarm algorithm has just enough moving parts to make it hard to understand. Then a gaussian version PSO named bare bones particle swarm optimization (BBPSO) is proposed, which may seem to be something other than particle swarms [17]. Different from the standard PSO, the paradigm in terms of particles “flying” through the search space appear more to splatter than fly. In BBPSO, the particle’s position is updated as follows:
Where $\mathbf{r}$ is the vector composed by random number which obeys $\mathcal{N}(0,1)$ normal distribution. BBPSO strips away some traditional features of the particle swarm in the search for the properties that make it work, and reveals some of the mysteries of the algorithm, and also performs better in some problems.

3. Proposed strategy

3.1. Selecting appropriate algorithms

As discussed above, different PSOs have different capabilities of exploitation and exploration, and each of these algorithms performs effective in some problems, but not in all problems. For comprehensive utilization capabilities of different algorithms to solve different problems, it can provide the following ideas: in the optimization process, the first thing is to judge whether the algorithm is appropriate to solve the problem, if it is, then use the algorithm, else judge another algorithm, until the appropriate algorithm is found. However, there is another question: which one is the appropriate algorithm? In the actual optimization problem, the optimal solution of the problem is always not known before the problem is solved, only it can obtain a better solution by comparing with each other, and the solution for comparison is also generated in the iterative process.

3.2. Multiple adaptive sub-swarms strategy

If an algorithm cannot search a better position, it should be replaced with another algorithm through a certain mechanism. In PSO, the initialization is important, which will greatly affect the final result. Different algorithms are used to optimize the same population at different stages, and the initial population for an algorithm is the final population for another one. If the population has been introduced to a local optimum by an algorithm, and other algorithm is difficult to guide the population jump out of the local trap.

Fig.1 Single swarm and multiple sub-swarms optimization

Fig.1 shows the difference between the single swarm strategy and the multiple sub-swarms strategy. Using multiple sub-swarm to optimize can avoid the local trap at the beginning. Moreover, parallel search can be used to improve the efficiency of the algorithm, because there is only the global information interrelated between populations. When using multiple swarms, each swarm can be optimized independently, and there is little interaction between swarms, so it can be carried out in parallel.

Although the parallel way can save a lot of time in the running when there are several cores in the computer. However, the multiple sub-swarms need more computing resources than the single swarm in total. Although after the same number of iterations, the multiple sub-swarms can achieve better results than the single group, but after the same number of function evaluations, multiple sub-swarms may get worse results than the single swarm. To this end, multiple sub-swarms adaptive strategy is proposed to reduce the consumption of computing resources.
Multi sub-swarms adaptive strategy initializes a large swarm, and then splits it into several sub-swarms. Each sub-swarm, sharing the same global best \( g \), uses different algorithms for computing in the iterative process. In some occasional cases after a certain number of iteration, the algorithm will fall into a state of temporary stagnation. If the search is no longer continues once stalled, then it is likely to miss a better solution. Therefore, it sets a positive integer constant \( RT \) here for each sub-swarm. Once the sub-swarm is trapped in a local optimum, there are \( RT \) chances to continue the search. However, if it still has not found a new better position after \( RT \) times of iteration, then stop the iteration of the sub-swarm to save the computing resources. Then, update \( g \) based on the iteration value of each sub-swarm. After a certain times of iteration, the chance of get a better position for each sub-swarm will be less and less, even all sub-swarms stop to converge. At this point, all sub-swarms will be restarted and given the opportunities of falling into stagnation to \( RT \) times. Repeat the steps above until it meets the stop criterion.

**Fig.2** Multiple sub-swarms adaptive strategy flow

Fig.2 shows the multiple sub-swarms adaptive strategy flow. The adaptive strategy not only can save computing resources, but also can be flexibly adapted to different types of problems by adjusting the value of \( RT \). In general, larger \( RT \) values are more suitable for complicated problems, while smaller \( RT \) values are more suitable for relatively simple problems.

**4. Experiments**

In order to obtain the comparison results, eight algorithms including PSOMAS are selected to optimize the benchmark functions. The algorithms and their control parameters are as follows:

- **FIPSO** (Full-informed PSO): \( N=20, \varphi=0.7298, c_1=c_2=2.05, \) Toplogy: Ring and Neighbors=3.
- **CPSO** (Cooperative PSO): \( N=20, w=0.7298, c_1=c_2=2.05 \) and \( S_t=6. \)
- **CLPSO** (Comprehensive learning PSO): \( N=20, w_{max}=0.9, w_{min}=0.4, c=1.494 \) and \( m=7. \)
- **AIWPSO** (Adaptive inertial weight PSO): \( N=20, w_{max}=1, w_{min}=0 \) and \( c_1=c_2=2.05. \)
• BBPSO (Bare bones PSO): N=20 and w=0.9.
• PSOMAS (PSO with multiple adaptive sub-swarms): N=100, N=20, RT=1, 10 or 100.

In PSOMAS, different sub-swarms are trained by different algorithms, the parameters of which is the same as each single algorithm.

According to the recommendation of problem definitions and evaluation criteria for the CEC2014 special session on real parameter optimization[18]. The above PSO algorithms use the same maximum number of fitness evaluations. For the 30 dimensional problems, it is set to 3E+5. All experiments were run independently for 30 times, and the average error and variance were recorded.

Tab. 1 presents the mean error values of the results achieved by the comparative algorithms. The best results among the comparative algorithms are shown in bold.

| Function | FIPS | CPSO | CLPSO | AIWPSO | BBPSO | PSOMAS | PSOMAS | PSOMAS |
|----------|------|------|-------|--------|-------|--------|--------|--------|
|          |      |      |       |        |       |       |       |        |
| $f_1$    | 6.29E+06 | 1.46E+07 | 1.38E+07 | 1.99E+07 | 1.96E+07 | 6.11E+06 | 4.69E+06 | 1.46E+06 |
| $f_2$    | 9.32E+03 | 5.48E+08 | 1.08E+04 | 7.05E+08 | 3.99E+09 | 1.20E+08 | 1.29E+03 | 9.06E-01 |
| $f_3$    | 4.01E+03 | 2.58E+04 | 1.14E+04 | 1.77E+04 | 2.25E+04 | 2.74E+03 | 4.33E+03 | 1.19E+03 |
| $f_4$    | 2.49E-03 | 2.01E+00 | 2.29E-03 | 7.91E-01 | 3.85E+00 | 7.24E-01 | 9.25E-02 | 4.27E-03 |
| $f_5$    | 2.09E+01 | 2.00E+01 | 2.07E+01 | 2.01E+01 | 2.07E+01 | 2.08E+01 | 2.07E+01 | 2.04E+01 |
| $f_6$    | 1.80E+01 | 1.80E+01 | 8.75E+00 | 2.75E+01 | 2.29E+01 | 2.01E+01 | 1.76E+01 | 1.47E+01 |
| $f_7$    | 3.77E-01 | 6.12E+00 | 2.03E-02 | 1.64E+01 | 2.10E+01 | 3.92E-01 | 1.84E-01 | 2.98E-02 |
| $f_8$    | 4.80E+03 | 9.53E+02 | 4.39E+01 | 1.33E+02 | 2.45E+03 | 1.08E+02 | 1.07E+02 | 7.14E+00 |
| $f_9$    | 1.14E+01 | 2.58E+01 | 4.50E+00 | 2.52E+01 | 3.57E+01 | 1.62E+01 | 1.02E+01 | 7.92E+00 |
| $f_{10}$ | 1.00E+04 | 1.00E+04 | 9.93E+03 | 1.05E+04 | 1.01E+04 | 8.80E+03 | 8.86E+03 | 9.60E+03 |
| $f_{11}$ | 9.71E+03 | 9.93E+03 | 9.80E+03 | 1.01E+04 | 9.75E+03 | 1.17E+03 | 1.76E+03 | 8.23E+03 |
| $f_{12}$ | 9.07E+05 | 9.30E+07 | 3.08E+03 | 6.97E+08 | 1.64E+07 | 3.34E+07 | 1.14E+07 | 4.36E+06 |
| $f_{13}$ | 2.77E-01 | 6.76E-01 | 2.71E-01 | 6.63E-01 | 5.68E-01 | 4.04E-01 | 3.72E-01 | 3.63E-01 |
| $f_{14}$ | 4.74E-01 | 1.56E+00 | 5.19E-01 | 2.84E+00 | 4.48E+00 | 3.61E-01 | 4.24E-01 | 3.73E-01 |
| $f_{15}$ | 6.67E+00 | 3.23E+02 | 1.34E+01 | 3.85E+02 | 3.35E+04 | 1.54E+01 | 1.37E+01 | 1.07E+01 |
| $f_{16}$ | 1.13E+01 | 1.21E+01 | 1.22E+01 | 1.23E+01 | 1.14E+01 | 1.26E+01 | 1.17E+01 | 1.15E+01 |
| $f_{17}$ | 3.16E+03 | 3.64E+03 | 3.20E+03 | 3.72E+03 | 4.08E+03 | 3.14E+03 | 3.12E+03 | 2.85E+03 |
| $f_{18}$ | 7.88E+04 | 3.52E+04 | 2.18E+03 | 3.20E+04 | 6.37E+04 | 2.25E+04 | 2.35E+04 | 3.86E+03 |
| $f_{19}$ | 3.18E+02 | 1.19E+02 | 5.16E+03 | 2.39E+06 | 6.41E+05 | 8.74E+02 | 1.26E+03 | 2.66E+02 |
| $f_{20}$ | 1.91E+04 | 6.31E+04 | 2.18E+04 | 9.81E+08 | 5.13E+08 | 5.26E+04 | 2.70E+04 | 5.85E+03 |
| $f_{21}$ | 8.97E+05 | 1.09E+06 | 2.00E+06 | 3.56E+06 | 4.57E+06 | 3.19E+05 | 1.43E+05 | 9.04E+04 |
| $f_{22}$ | 2.52E+03 | 1.07E+16 | 5.51E+10 | 2.64E+14 | 1.20E+09 | 3.43E+03 | 2.38E+03 | 1.94E+03 |
| $f_{23}$ | 3.29E+02 | 3.17E+02 | 3.19E+02 | 3.16E+02 | 3.30E+02 | 3.12E+02 | 3.08E+02 | 3.09E+02 |
| $f_{24}$ | 3.31E+03 | 4.65E+03 | 4.33E+03 | 5.05E+03 | 6.24E+03 | 4.35E+03 | 4.34E+03 | 4.34E+03 |
| $f_{25}$ | 3.16E+03 | 2.88E+03 | 2.57E+03 | 4.27E+03 | 2.89E+03 | 2.96E+02 | 2.91E+02 | 8.64E+02 |
| $f_{26}$ | 8.64E+02 | 8.80E+02 | 8.68E+02 | 9.51E+02 | 9.04E+02 | 8.71E+02 | 8.71E+02 | 8.66E+02 |
| $f_{27}$ | 9.26E+02 | 7.71E+02 | 8.72E+02 | 9.53E+02 | 8.02E+02 | 8.06E+02 | 7.82E+02 | 7.48E+02 |
| $f_{28}$ | 9.07E+02 | 9.14E+02 | 8.91E+02 | 1.31E+03 | 1.02E+03 | 7.40E+02 | 6.42E+02 | 7.03E+02 |
| $f_{29}$ | 7.23E+07 | 2.21E+08 | 6.10E+07 | 3.06E+08 | 2.15E+08 | 1.39E+07 | 1.20E+07 | 4.38E+07 |
| $f_{30}$ | 4.84E+08 | 6.29E+16 | 8.33E+11 | 6.01E+15 | 2.62E+11 | 1.90E+09 | 1.91E+10 | 1.17E+08 |

Tab. 1 Comparison results of mean error values
From the results, we can see that PSOMAS ($RT=100$) obtains the minimum average error value on 11 functions, while PSOMAS with $RT=10$ and $RT=1$ obtain best results on 4 and 3 functions, respectively. Among the rest algorithms, CLPSO has the smallest average error on 7 functions, while FIPSO and CPSO get the best results on 4 and 1 functions respectively. The experiments results demonstrate that PSOMAS (especially with the $RT=100$) could find higher quality solutions than the other algorithms on the 30 benchmark functions. More specifically, PSOMAS is more suitable to optimize the unimodal, hybrid and composition functions, while CLPSO is more suitable to optimize the simple multimodal functions. The comparison results also show that PSOMAS is much better with the larger $RT$ value of on these benchmark functions.

5. Conclusion
From the perspective of enhancing exploration and exploitation capabilities of the algorithms, this paper proposes a particle swarm optimization with multiple adaptive sub-swarms, which adaptively adopts different search algorithms in the iterative process. This method dynamically selects different sub-swarm with different PSO training according to their previous successful searching memories. Thus, this proposed algorithm takes advantage of single PSOs being good at different complicated optimizations, then can adapt to more problems. Extensive experimental results on the CEC 2014 benchmark function suite validated by statistical significance analysis demonstrate that the proposed method is very effective. Moreover, different $RT$ values make the proposed algorithm is able to adapt to different problems. In these experiments above, the bigger $RT$ value makes the algorithm get a better result.

In the proposed hybrid framework, there are many other interesting and powerful search strategies to achieve better performance. Many other better PSOs can be tried for the sub-swarm training. How to select and combine PSOs will be investigated in the future work.

References
[1] Clerc M, Kennedy J 2002 The particle swarm - explosion, stability, and convergence in a multi-dimensional complex space J. IEEE Trans. on Evolutionary Computation 6 pp 58-73
[2] Yin R, Li Q, Li P and et al 2020 Parameter identification of multistage fracturing horizontal well based on PSO-RBF Neural Network entific Programming pp 1-11
[3] Kumar M, Sharma S C 2019 PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing Neural Computing and Applications
[4] Deng W, Yao R, Zhao H and et al 2017 A novel intelligent diagnosis method using optimal LS-SVM with improved PSO algorithm Soft Comput.
[5] Yuan J 2020 An Anomaly data mining method for mass sensor networks using improved PSO algorithm based on spark parallel framework J. Grid Computing 3
[6] Liu B, Li J, Lin W, et al 2020 K-PSO: An improved PSO-based container scheduling algorithm for big data applications Int. J. Network Management 1
[7] Marini F, Walczak B 2015 Particle swarm optimization (PSO) a tutorial Chemometrics & Intelligent Laboratory Systems 149 pp 153-165
[8] Parouha R P, Das K N 2015 Parallel hybridization of differential evolution and particle swarm optimization for constrained optimization with its application. Int. J. System Assurance Engineering & Management pp 1-20
[9] Novoa-Hernández P, Corona C C and Pelta D A 2011 Efficient multi-swarm PSO algorithms for dynamic environments Memetic Comput. 3(3) pp 163-174
[10] Wolpert D H, Macready W G 1997 No Free Lunch Theorems for Optimization IEEE Trans. on Evolutionary Computation 1 pp 67-82
[11] Pornsing C, Sodhi M S and Lamond B F 2015 Novel self-adaptive particle swarm optimization methods Soft Comput pp 1-15
[12] Mendes R, Kennedy J, Neves J 2004 The fully informed particle swarm: simpler, maybe better Evolutionary ComputationIEEE Trans. on 8(3) pp 204-210
[13] Frans V D B, Engelbrecht A P 2004 A Cooperative approach to particle swarm optimization. *IEEE Trans. on Evolutionary Computation* **8**(3) pp 225-239

[14] Liang J J, Qin A K and Suganthan P N 2006 Comprehensive learning particle swarm optimizer for global optimization of multimodal functions *Evolutionary Computation IEEE Trans. On* **10**(3) pp 281-295

[15] Yu X, Zhang X 2014 Enhanced comprehensive learning particle swarm optimization *Appl. Mathematics & Computation* **242** pp 265-276

[16] Nickabadi A, Ebadzadeh M M and Safabakhsh R 2011 A novel particle swarm optimization algorithm with adaptive inertia weight *Appl. Soft Comput* **11**(4) pp 3658-3670.

[17] Kennedy J 2003 Bare bones particle swarms. Swarm Intelligence Symposium SIS ’03. *Proc. the 2003 IEEE. IEEE 2003* pp 80-87

[18] Liang J J, Qu B Y and Suganthan P N 2013 Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization *Technical Report*