A Parallel Strategy Applied to APSO

Qing-Wei Chai, Jeng-Shyang Pan, Wei-Min Zheng, and Shu-Chuan Chu*

Department of Computer Science and Engineering,
Shandong University of Science and Technology,
Qingdao 266590,China
scchu0803@gmail.com

Abstract. *Particle Swarm Optimization(PSO) is a famous and effective branch of evolutionary computation, which aims at tackling complex optimization problems. Parallel strategy is an excellent method which separate the population into some subgroups, the subgroups can communicate with each other to improve algorithms' performance significantly. In this paper, we apply a parallel method on Adaptive Particle Swarm Optimization(APSO), to further improve convergence speed and global search ability of Parallel PSO. The novel Parallel APSO algorithm was verified under many benchmarks of the Congress on Evolutionary Computation(CEC) Competition test suites on real-parameter single-objective optimization and the experimental results showed the proposed Parallel APSO algorithm was competitive with the Parallel PSO.*

Keywords: Parallel PSO, APSO, Parallel APSO

1 Introduction

Evolutionary computation has been gotten great progress with many aspects in recent years, thus on many real-world optimization problems, evolutionary computation is used more and more frequently. For example, sensor ontology meta-matching[1], demand estimation[2] and wireless sensor network[3] have successfully applied evolutionary computing to improve performance. Many methods also have been proposed in recent years, such as parameter adaptive adjust[4], QUATRE[5–10] and new estimation strategy[11]. Many parallel schemes have been applied to swarm intelligence algorithm[12–18].

Mimic the swarm behaviors of birds flocking and fish schooling, Kennedy and Eberhart introduced the PSO firstly in 1995[19]. In PSO, a swarm of particles is represented as potential solutions, and each particle i is associated with two vectors, i.e., the velocity vector $V_i = \{V_{i1}, V_{i2}, ..., V_{iD}\}$ and the position vector $X_i = \{x_{i1}, x_{i2}, ..., x_{iD}\}$. During the evolutionary process, the velocity and position of particle i are updated as follows

$$v_i^{t+1} = w^t \cdot v_i^t + c_1 \cdot r_1 \cdot (pBest_i^t - x_i^t) + c_2 \cdot r_2 \cdot (gBest^t - x_i^t) \tag{1}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, i = 1, 2, \ldots, N \tag{2}$$

* Corresponding author

Where t stands for the number of iteration, N is the population size, r_1 and r_2 are two random numbers between 0 and 1, $pBest_i^t$ is the best value for the i_{th} particle at t iteration and $gBest$ is the best value for all particles until current iteration. As the adjustment of parameters can significantly influence the performance of PSO, moderately adjust parameters is an important task to improve the performance of PSO. Zhan et al. proposed an improved PSO version which can adjust parameters adaptively to change search ability[20]. Other more, a parallel strategy is proposed by Chang et al.[21] for separating particles into some subgroups to search optimal value. After running some iterations, the subgroups communicate together by exchanging optimal information among subgroups to accelerate convergence rate and enhance global search ability.

2 Review of APSO and Parallel PSO

In this part, several famous and powerful PSO variants are reviewed briefly, such as Parallel PSO and APSO. They improve the performance of PSO algorithm greatly in terms of convergence speed and search ability, they all closely related to the proposed Parallel APSO algorithm in the paper. By reviewing these former state-of-the-art PSO variants, researchers can get a better understanding of why and how we proposed the Parallel APSO algorithm.

2.1 Adaptive Particle Swarm Optimization

Although PSO has been commonly implemented in recent years, it has two significant concerned necessarily problems that accelerate convergence speed and avoiding the local optima. Zhan et at.[20] proposed an approach with a global search ability for faster convergence speed. The parameter can be fitted properly by evaluating the distribution state and fitness value of particles during optimal process. The distribution state consists of the following four defined evolutionary states including exploration, exploitation, convergence, and jumping out. It guarantees the automatic control of inertia weight, acceleration coefficients, and other algorithmic parameters at running time to enhance the search ability and convergence speed. Then, an elitist learning strategy is applied when particles trapped into a local optimal state. The strategy will act on the globally best particle to jump out of the likely local optima. Therefore, the algorithm will take into consideration the population distribution state for each generation, as detailed in the following steps.

1. At the current position, calculate the mean distance of each particle i to all the other particles. For example, this mean distance of the i_{th} particle d_i can be measured using an Euclidian metric.

$$d_i = \frac{1}{N-1} \sum_{j=1, j \neq i}^{N} \sqrt{\sum_{k=1}^{D} (X_i^k - X_j^k)^2} \tag{3}$$

Where N and D are the population size and the number of dimensions, respectively.

2. d_i is denoted as the globally best particle d_g. Compare all d_i's, and determine the maximum and minimum distances d_{max} and d_{min}. Compute an "evolutionary factor" f as defined by

$$f = \frac{d_g - d_{min}}{d_{max} - d_{min}} \in [0, 1] \tag{4}$$

3. Classify f into one of the four sets $S1$, $S2$, $S3$, and $S4$, which represent the states of exploration, exploitation, convergence, and jumping out, respectively.

When the status of PSO is determined by analyzing the population distribution information, the algorithm will adjust the parameters w, c_1 and c_2. The details of adjusting are presented in $Eq.$5 and Table 1 respectively:

$$w(f) = \frac{1}{1 + 1.5e^{-2.6f}} \in [0.4, 0.9], \forall f \in [0, 1] \tag{5}$$

Table 1. STRATEGIES FOR THE CONTROL COEFFICIENTS OF C_1 AND C_2

State	Strategy	C_1	C_2
Exploration	S1	Increase	Decrease
Exploitation	S2	Increase Slightly	Decrease Slightly
Convergence	S3	Increase Slightly	Increase Slightly
$Jumping - out$	S4	Decrease	Increase

2.2 Parallel PSO

Chang et al.[21] proposed three parallel strategies for PSO in Parallel PSO algorithm. The new algorithm aims to produce better results achievable using only one processor with the goal of reducing the running time. The Parallel PSO significantly improves the rate of convergence.

Parallel PSO algorithm uses three communication strategy to exchange optimal information among subgroups. If parameters are independent or are only loosely correlated, the first strategy is adopted. As the better particles are more promising to obtain the best result, multiple copies of the best particles are mutated, and these mutated particles replace the poorer particles in the other subgroups during some fixed iterations.

If the parameters of a solution are loosely correlated, in this case, the better particles in each group have little probability to obtain optimum results quickly. The second communication strategy may be more proper to be applied. The best particle in each group is migrated to its neighboring groups to replace some

more poorly performing particles every some fixed iterations. Since the number of subgroups can be defined as a power of two, neighbors are defined as those subgroups whose binary representations can be differed by one bit.
When we don't know the correlation property of the solution space, the first and second communication strategies can't work well. As a result, a hybrid communication strategy is implemented. The groups are separated into two-equal sized subgroups. One-half groups apply the first strategy every some fixed iteration and all groups applying the second strategy every some fixed iteration.

3 Parallel APSO

As mentioned above, APSO adjusts the equation's parameters to improve the convergence rate of the original PSO. Parallel PSO enhance the ability of PSO by exchanging optimum information between groups. In this paper, we implement parallel strategy on APSO to improve the convergence rate of APSO and entirely explore test zone.
In order to show the advantages of Parallel APSO, we choose 28 benchmark functions from CEC2013 to test the performance of Parallel APSO and compare with the results obtained by PSO, Parallel PSO and APSO. In order to make a fair comparison among the algorithms, the parameter configurations of different PSOs are set the same. The inertia weight w is initialized to 0.9 and linearly decrease to 0.4 at the last iteration, c_1 and c_2 set to 2.0, they use the same population size of 40, it is the same as the common configuration in a standard PSO. The Parallel PSO and new proposed algorithm separate particles to four groups, as the correlation property of fitness function is unknown, this paper adopts the third communication strategy mentioned above.
For the purpose of reducing statistical errors, each function is simulated 30 times under 30, 40 and 50 dimensions respectively, and their mean results are used for the comparison. We can see the detailed comparison of these algorithms in Table 2.
The experimental results in Table 1 contain the mean fitness of the 30 trials on each function with different PSO algorithms. The results show that PAPSO has a stronger ability to reach the global optimum on most of the tested functions. The new algorithm obtained 13,20,15 best results under 30D, 40D, 50D fitness function respectively. Figure 1 to Figure 6 show the specific process of optimization under F8 and F20 in 30 dimensions, F20 and F23 in 40 dimensions, F8 and F23 in 50 dimensions respectively.

Table 2. Mean value got from running 30 times under 30D,40D,50D fitness function

Functions	PPSO			APSO			PAPSO		
Dimension	30D	40D	50D	30D	40D	50D	30D	40D	50D
f1	-1.40×10^3	-1.40×10^3	-1.40×10^3	-1.40×10^3	-1.40×10^3	-1.40×10^3	-1.40×10^3	-1.40×10^3	-1.40×10^3
f2	6.17×10^6	1.08×10^7	2.03×10^7	5.42×10^5	2.24×10^5	2.38×10^6	5.94×10^5	3.38×10^5	2.43×10^6
f3	1.76×10^9	1.81×10^9	4.80×10^9	7.12×10^8	7.06×10^8	2.07×10^9	2.84×10^8	5.14×10^8	1.03×10^9
f4	1.44×10^3	4.97×10^3	8.59×10^3	4.02×10^3	1.09×10^3	1.51×10^4	4.17×10^3	2.11×10^3	1.27×10^4
f5	-9.97×10^2	-9.84×10^2	-9.41×10^2	-1.00×10^3	-1.00×10^3	-1.00×10^3	-1.00×10^3	-1.00×10^3	-1.00×10^3
f6	-8.40×10^2	-7.93×10^2	-7.97×10^2	-8.62×10^2	-8.15×10^2	-8.37×10^2	-8.58×10^2	-8.14×10^2	-8.26×10^2
f7	-6.60×10^2	-6.70×10^2	-6.70×10^2	-2.24×10^2	-6.07×10^2	-2.96×10^2	-6.49×10^2	-6.83×10^2	-6.34×10^2
f8	-6.79×10^2	-6.79×10^2	-6.79×10^2	-6.79×10^2	-6.80×10^2	-6.79×10^2	-6.79×10^2	-6.80×10^2	-6.79×10^2
f9	-5.68×10^2	-5.59×10^2	-5.40×10^2	-5.61×10^2	-5.63×10^2	-5.27×10^2	-5.64×10^2	-5.65×10^2	-5.32×10^2
f10	-4.92×10^2	-4.82×10^2	-4.57×10^2	-5.00×10^2	-5.00×10^2	-4.99×10^2	-5.00×10^2	-5.00×10^2	-4.99×10^2
f11	-4.47×10^2	-4.83×10^2	7.52×10^1	-3.96×10^2	-3.96×10^2	-3.96×10^2	-3.96×10^2	-3.96×10^2	-3.97×10^2
f12	-3.63×10^1	9.22×10^1	1.86×10^2	1.97×10^2	3.43×10^2	4.83×10^2	1.19×10^2	2.69×10^2	4.58×10^2
f13	1.52×10^2	3.30×10^2	4.79×10^1	3.37×10^2	5.17×10^2	7.50×10^2	3.31×10^2	4.78×10^2	6.55×10^2
f14	3.66×10^3	5.27×10^3	7.15×10^3	1.39×10^3	1.69×10^3	2.12×10^3	1.25×10^3	1.61×10^3	2.14×10^3
f15	4.51×10^3	6.20×10^3	8.78×10^3	5.02×10^3	4.75×10^3	8.85×10^3	4.52×10^3	4.38×10^3	8.16×10^3
f16	2.01×10^2	2.02×10^2	2.02×10^2	2.01×10^2	2.00×10^2	2.02×10^2	2.01×10^2	2.00×10^2	2.02×10^2
f17	5.26×10^2	6.56×10^2	8.51×10^2	3.30×10^2	3.41×10^2	3.51×10^2	3.30×10^2	3.41×10^2	3.51×10^2
f18	6.46×10^2	8.05×10^2	9.58×10^2	7.98×10^2	9.53×10^2	1.20×10^3	7.51×10^2	8.65×10^2	1.14×10^3
f19	5.14×10^2	5.23×10^2	5.36×10^2	5.01×10^2	5.01×10^2	5.02×10^2	5.01×10^2	5.01×10^2	5.02×10^2
f20	6.15×10^2	6.18×10^2	6.25×10^2	6.15×10^2	6.17×10^2	6.24×10^2	6.15×10^2	6.16×10^2	6.24×10^2
f21	1.03×10^3	1.40×10^3	1.62×10^3	1.02×10^3	1.31×10^3	1.56×10^3	1.05×10^3	1.41×10^3	1.54×10^3
f22	5.70×10^3	8.80×10^3	1.17×10^4	2.65×10^3	3.58×10^3	4.47×10^3	2.49×10^3	3.37×10^3	4.16×10^3
f23	6.61×10^3	9.27×10^3	1.22×10^4	7.02×10^3	7.62×10^3	1.22×10^4	6.49×10^4	6.57×10^3	1.15×10^4
f24	1.29×10^3	1.33×10^3	1.38×10^3	1.34×10^3	1.32×10^3	1.46×10^3	1.31×10^3	1.31×10^3	1.42×10^3
f25	1.43×10^3	1.49×10^3	1.54×10^3	1.49×10^3	1.50×10^3	1.66×10^3	1.46×10^3	1.48×10^3	1.62×10^3
f26	1.48×10^3	1.41×10^3	1.52×10^3	1.58×10^3	1.54×10^3	1.69×10^3	1.49×10^3	1.43×10^3	1.50×10^3
f27	2.47×10^3	2.83×10^3	3.34×10^3	2.74×10^3	2.85×10^3	3.93×10^3	2.65×10^3	2.65×10^3	3.59×10^3
f28	3.02×10^3	4.38×10^3	3.78×10^3	5.82×10^3	3.58×10^3	9.13×10^3	5.38×10^3	3.47×10^3	8.58×10^3

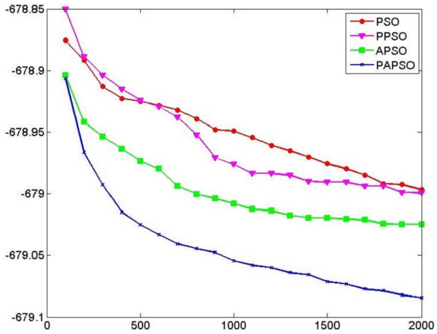

Fig. 1. $30_D, F_8$

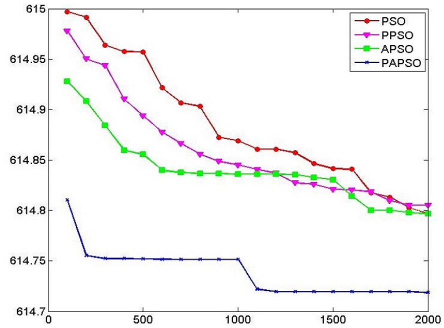

Fig. 2. $30_D, F_{20}$

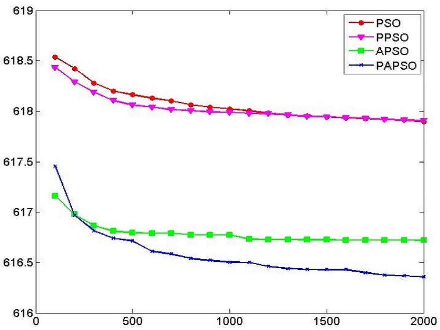

Fig. 3. $40_D, F_{20}$

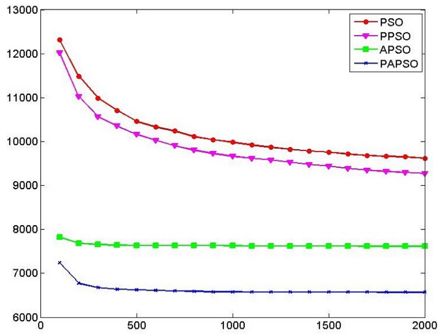

Fig. 4. $40_D, F_{23}$

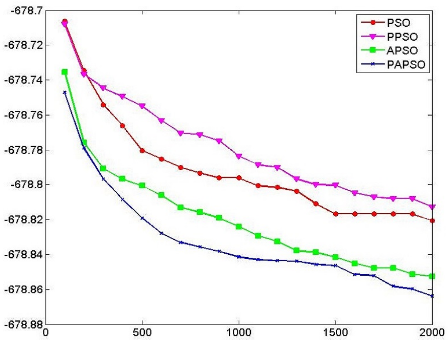

Fig. 5. $50_D, F_8$

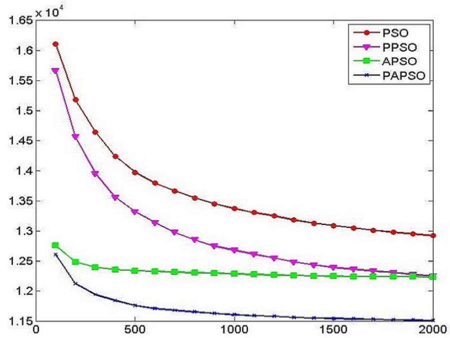

Fig. 6. $50_D, F_{23}$

4 Conclusion

This paper introduces the Parallel PSO algorithm and APSO briefly, then apply a useful parallel strategy on APSO, by separating the population into some subgroups to obtain more optimal information than APSO. Though adjustment parameter is an important method for improving the functioning of PSO and other evolutionary computation, complex computation is a fatal flaw limit its application for real-time problem. From the data of table and figures, we can clearly see the novel algorithm has more quickly convergence rate under some fitness function. In some other fitness function, the novel algorithm has the strongest search ability. Experimental results show the parallel strategy is an useful and simple method to enhance the ability of APSO algorithm.

References

1. Xingsi Xue and Jeng-Shyang Pan. A compact co-evolutionary algorithm for sensor ontology meta-matching. *Knowledge and Information Systems*, 56(2):335–353, 2018.
2. Hui Wang, Wenjun Wang, Zhihua Cui, Xinyu Zhou, Jia Zhao, and Ya Li. A new dynamic firefly algorithm for demand estimation of water resources. *Information Sciences*, 438:95–106, 2018.
3. Jeng-Shyang Pan, Lingping Kong, Tien-Wen Sung, Pei-Wei Tsai, and Václav Snášel. A clustering scheme for wireless sensor networks based on genetic algorithm and dominating set. *Journal of Internet Technology*, 19(4):1111–1118, 2018.
4. Zhenyu Meng, Jeng-Shyang Pan, and Lingping Kong. Parameters with adaptive learning mechanism (palm) for the enhancement of differential evolution. *Knowledge-Based Systems*, 141:92–112, 2018.
5. Zhenyu Meng, Jeng-Shyang Pan, and Huarong Xu. Quasi-affine transformation evolutionary (quatre) algorithm: a cooperative swarm based algorithm for global optimization. *Knowledge-Based Systems*, 109:104–121, 2016.
6. Zhenyu Meng and Jeng-Shyang Pan. A competitive quasi-affine transformation evolutionary (c-quatre) algorithm for global optimization. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 001644–001649. IEEE, 2016.
7. Jeng-Shyang Pan, Zhenyu Meng, Huarong Xu, and Xiaoqing Li. Quasi-affine transformation evolution (quatre) algorithm: A new simple and accurate structure for global optimization. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 657–667. Springer, 2016.
8. Jeng-Shyang Pan, Zhenyu Meng, Shu-Chuan Chu, and John F Roddick. Quatre algorithm with sort strategy for global optimization in comparison with de and pso variants. In *The Euro-China Conference on Intelligent Data Analysis and Applications*, pages 314–323. Springer, 2017.
9. Zhenyu Meng and Jeng-Shyang Pan. Quasi-affine transformation evolutionary (quatre) algorithm: a parameter-reduced differential evolution algorithm for optimization problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4082–4089. IEEE, 2016.
10. Zhenyu Meng and Jeng-Shyang Pan. Quasi-affine transformation evolutionary (quatre) algorithm: the framework analysis for global optimization and application in hand gesture segmentation. In *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, pages 1832–1837. IEEE, 2016.
11. Chaoli Sun, Jianchao Zeng, Jengshyang Pan, Songdong Xue, and Yaochu Jin. A new fitness estimation strategy for particle swarm optimization. *Information Sciences*, 221:355–370, 2013.

12. Thi-Kien Dao, Tien-Szu Pan, Trong-The Nguyen, and Jeng-Shyang Pan. Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *Journal of Intelligent Manufacturing*, 29(2):451–462, 2018.
13. Van-Oanh Sai, Chin-Shiuh Shieh, Trong-The Nguyen, Yuh-Chung Lin, Mong-Fong Horng, and Quang-Duy Le. Parallel firefly algorithm for localization algorithm in wireless sensor network. In *2015 Third International Conference on Robot, Vision and Signal Processing (RVSP)*, pages 300–305. IEEE, 2015.
14. Cheng-Fu Tsai, Thi-Kien Dao, Wei-Jie Yang, Trong-The Nguyen, and Tien-Szu Pan. Parallelized bat algorithm with a communication strategy. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 87–95. Springer, 2014.
15. Trong-The Nguyen, Chin-Shiuh Shieh, Mong-Fong Horng, Thi-Kien Dao, and Truong-Giang Ngo. Parallelized flower pollination algorithm with a communication strategy. In *2015 Seventh International Conference on Knowledge and Systems Engineering (KSE)*, pages 103–107. IEEE, 2015.
16. Pei-Wei Tsai, Jeng-Shyang Pan, Shyi-Ming Chen, and Bin-Yih Liao. Enhanced parallel cat swarm optimization based on the taguchi method. *Expert Systems with Applications*, 39(7):6309–6319, 2012.
17. Jeng-Shyang Pan, F.R. Mcinnes, and M.A Jack. Application of parallel genetic algorithm and property of multiple global optima to vq codevector index assignment for noisy channels. *Electronics Letters*, 32(4):296–297, 1996.
18. Shu-Chuan Chu, John F. Roddick, and Jeng-Shyang Pan. Ant colony system with communication strategies. *Information Sciences*, 167(1-4):63–76, 2004.
19. Jmaes Kennedy and Russell Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks, Perth, Australia*, pages 1942–1948, 1995.
20. Zhi-Hui Zhan, Jun Zhang, Yun Li, and Henry Shu-Hung Chung. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1362–1381, 2009.
21. Jui-Fang Chang, Shu-Chuan Chu, John F. Roddick, and Jeng-Shyang Pan. A parallel particle swarm optimization algorithm with communication strategies. 2005.