Journal of Computers Vol. 31 No. 1, 2020, pp. 85-93
doi:10.3966/199115992020023101007

An Improved Particle Swarm Optimization Algorithm Based on Restart Strategy

Hu Huang[1], Yu-Hui Lei[1*], Chen-Hao Xiong[2], Ding Yang[1]

[1] College of Information Science & Technology, Chengdu University of Technology, Chengdu 610059, Sichuan, China
hhu@cdut.edu.cn, sakura_ray@qq.com, 1170951913@qq.com

[2] College of Energy, Chengdu University of Technology, Chengdu 610059, Sichuan, China
treexch@126.com

Received 7 May 2018; Revised 26 October 2018; Accepted 18 November 2018

Abstract. In order to solving the problems of premature convergence and local extremum caused by traditional particle swarm optimization algorithm (PSO algorithm), this paper proposes an improved particle swarm optimization algorithm based on restart strategy (BRSPSO algorithm). In the process of optimization, BRSPSO algorithm can update the speed and location of individual particles, increase the influence of excellent particles according to the mutual influence between particles. It introduces a restart strategy, at the same time, to enlarge the search scope of the algorithm and enhance the searching ability for the sake of preventing the search from falling into the local extremum. In this paper, BRSPSO algorithm is used to optimize the parameters of the support vector machine (SVM). Experiments show that compared with SVM optimized by traditional PSO algorithm and SVM without PSO, SVM optimized by BRSPSO algorithm can expand the search space and effectively avoid the local convergence problem of the traditional PSO algorithm.

Keywords: particle swarm optimization, restart strategy, swarm intelligence optimization

1 Introduction

The Particle Swarm Optimization algorithm (PSO algorithm), also known as the Birds Group Foraging Algorithm, is a novel swarm intelligence optimization algorithm proposed by Kennedy and Eberhart [1] in 1995. The PSO algorithm is a simulation of the foraging behavior of birds, starting from random particles, it finds the optimal solution through iteration. The advantage of the PSO algorithm is that the basic principle is simple, the convergence speed is fast, and it has less adjustment parameters. Currently, it has been widely used in multi-objective function optimization [2], neural network training [3] and other fields. But the PSO algorithm also has the disadvantages of easy premature convergence and local optimality. In response to such problems, many scholars have proposed some improved algorithms in recent years. Jiang et al. [4] proposed a multi-dimensional inertia weight-attenuation chaotic particle swarm optimization algorithm. In the process of particle swarm evolution, each dimension of each particle in each generation group is given a different linear attenuation chaotic inertia weight, which is enhanced. The particle activity and local search ability of the particle in the late search. Zhu et al. [5] proposed an improved particle swarm optimization algorithm using exponentially decreasing inertia weighting strategy, which overcomes the shortcomings of the fixed inertia weight and adaptive inertia weight particle swarm algorithm which is easy to fall into local minima and is applied to radial basis. Soft-sensor modeling method for MP fermentation process based on neural network. Luo et al. [6] proposed a genetic-particle swarm synthesis algorithm with dual learning ability by introducing genetic algorithm and incorporating oscillation parameter strategy, and applied it to the optimization design of

* Corresponding Author

stator double-winding wind-induced generator. Liu et al. [7] proposed an improved K-means clustering algorithm based on particle swarm optimization based on the characteristics of fusion data. The algorithm optimizes the weight coefficient in the optimization process, and adjusts the convergence speed as the number of iterations increases. Search is more balanced. The above method improves the traditional PSO algorithm by directly modifying or dynamically changing the value of a parameter in the PSO algorithm, or by combining with other algorithms, and does not consider the interaction between individual particles in the optimization process, and the algorithm has a limited search range and is difficult to overcome. The PSO algorithm is easy to fall into the disadvantage of local extremum.

Therefore, this paper analyzes the mutual influence of particles in the process of optimization based on the predecessors. By improving the influence of excellent particles and introducing the restart strategy, an improved particle swarm optimization algorithm based on restart strategy (BRSPSO) is proposed. In this paper, the parameters of support vector machine are optimized by BRSPSO algorithm. Finally, the simulation experiment proves that BRSPSO algorithm expands the search space compared with traditional PSO algorithm and does not use particle swarm optimization algorithm, which effectively avoids the local convergence problem of tradition-al PSO algorithm.

2 The Traditional PSO Algorithm

The traditional PSO algorithm is an iterative-based optimization algorithm. Each of the optimized particles is imagined as a bird, which is distributed in the *N*-dimensional space, and each particle has a fitness function to judge the current position. All particles find the optimal solution in the *N*-dimensional space according to fitness score. The specific implementation steps are as follows, assuming that there are *m* particles in the *N*-dimensional space, each particle is given a memory function, and can remember the best position that it has ever been. $v_i = (v_{i1}, v_{i2} \cdots, v_{iN})$ indicates the velocity of the *i*-th particle, $p_i = (p_{i1}, \cdots, p_{iN})$ indicates the position of the *i*-th particle, $pl_i = (pl_{i1}, \cdots, pl_{iN})$ indicates the best position of the history that *i*-th particle has experienced, $pg = (pg_1, \cdots, pg_N)$ indicates the global optimal position in the particle swarm.

In the *t*+1th iteration, the *i*-th particle velocity and position are updated in the *n*-dimensional by the formula (1) and (2).

$$v_{in}^{(t+1)} = \omega v_{in}^{(t)} + c_1 r_1 \left(pl_{\text{in}} - p_{in}^{(t)} \right) + c_2 r_2 \left(pg_n - p_{in}^{(t)} \right) \tag{1}$$

$$p_{in}^{(t+1)} = p_{in}^{(t)} + v_{in}^{(t+1)} \tag{2}$$

In the formula (1), $\omega \in [\omega_{\min}, \omega_{\max}]$, $v \in [v_{\min}, v_{\max}]$, ω is called as inertia weight, used for regulating the searching ability of the solution space. $\omega_{\min}$ and $\omega_{\max}$ are the minimum and maximum values of ω respectively, $v_{\min}$ and $v_{\max}$ are the minimum and maximum values of v respectively. r_1 and r_2 are the random numbers distributed uniformly between [0, 1], which can increase search randomness. c_1 and c_2 are called learning factors, used for adjusting the maximum step size of learning. Formula (1) can be understood as three parts, $\omega v_{in}^{(t)}$ indicates the inertia of the particle, ω indicates the tendency of particles to maintain the state of motion at the last moment. $c_1 r_1 \left(pl_{in} - p_{in}^{(t)} \right)$ indicates the influence of particle's own experience on particle motion, $c_2 r_2 \left(pg_{\text{n}} - p_{in}^{(t)} \right)$ indicates the influence of particle's social learning experience on particles.

The main problem of the traditional PSO optimization algorithm is that it is easy to prematurely converge and easy to fall into local optimum. It is mainly due to the fact that in the iterative process, the descendants of the particles will be clustered in the neighborhood of the particle history and the global optimal particle [5], and the diversity of the particles is gradually reduced in the later search [8], so this paper proposes an improved particle optimization algorithm based on restart strategy.

3 BRSPSO Algorithm

In order to ensure the searching ability of the algorithm, it is necessary to consider the mutual learning between particles, that is, the particle should influence the position and speed of the next time of each particle and increase the influence of the excellent particles.

Set $F\left(x_i^{(t)}\right)$ be the fitness function of the *t*-th particle at the *t*-th iteration, $avgF^{(t)} = \left(\sum_{i=1}^{m} F\left(x_i^{(t)}\right)\right)/m$ be the average fitness for the *t*-th iteration of the particles. We divide the iterations of *i*-th particle into three cases: the first iteration is state I, in the *t*-th iteration ($2 \le t \le T_{\max}$, $T_{\max}$ is the maximum number of iterations), when $F\left(x_i^{(t)}\right) < \text{avg}F^{(t)}$ and $avgF^{(t)} < avgF^{(t\text{-}1)}$ is state II, at this point, x_i is an excellent particle, when $F\left(x_i^{(t)}\right) > \text{avg}F^{(t)}$ and $avgF^{(t)} > avgF^{(t\text{-}1)}$ is state III, at this point, x_i is an bad particle, other conditions are considered as state IV。

The setting of the inertia weight ω is generally determined by the empirical method. It is generally set as a fixed value, but the larger the inertia weight, the better the global search, the smaller the better the local search [9]. The ideal algorithm should have good global search and local search ability at the same time. In the BRSPSO algorithm, the value of ω in state I, II, III, IV are updated by the formula (3):

$$\omega = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min}) * F\left(x_i^{(t)}\right) * (\theta + 1)}{\theta_{\max}} \tag{3}$$

In formula (3), $\theta_{\max}$ is the maximum of repetitions for the optimization value allowed, $\theta_{\max} \in [\theta_{\min}, \theta_{\max}]$, $\theta_{\min}$ and $\theta_{\max}$ are the minimum and maximum of θ.

Learning factors c_1 and c_2 indicates the own experience and social experience of the particles, The lower learning factor will make the particles in the neighborhood, although it can prevent the calculation spillover, it cannot explore enough, which may fall into the local extremum area. And the higher value can make the particle avoid falling into the local extremum, but there is a great probability to miss the target area. Therefore, the inertia weight ω, c_1 and c_2 must be carefully set up. c_1 and c_2 do not necessarily have to be equal to 2, but equal to a constant value can get a better solution [10].

In the BRSPSO algorithm, set $F\left(x_i^{(t)}\right)\_\max$ is the maximum value of $F\left(x_i^{(t)}\right)$ and $|1 - F\left(x_i^{(t)}\right)|$, $F\left(x_i^{(t)}\right)\_\min$ is the minimum value of $F\left(x_i^{(t)}\right)$ and $|1 - F\left(x_i^{(t)}\right)|$, c_1 and c_2, r_1 and r_2 are updated by the formula (4), (5) and (6) in state I, II, III, IV.

In state I, IV,

$$\begin{cases} r_1 = F\left(x_i^{(t)}\right) \\ r_2 = F\left(x_i^{(t)}\right) \\ c_1 = \left[\varepsilon * F\left(x_i^{(t)}\right) + 0.5\right] \\ c_2 = \left[\varepsilon * F\left(x_i^{(t)}\right) + 0.5\right] \end{cases} \tag{4}$$

In state II,

$$\begin{cases} r_1 = F\left(x_i^{(t)}\right)\_\max \\ r_2 = F\left(x_i^{(t)}\right)\_\min \\ c_1 = \left[\varepsilon * F\left(x_i^{(t)}\right)\_\max + 0.5\right] \\ c_2 = \left[\varepsilon * F\left(x_i^{(t)}\right)\_\min + 0.5\right] \end{cases} \tag{5}$$

In state III:

$$\begin{cases} r_2 = F\left(x_i^{(t)}\right)\_\max \\ r_1 = F\left(x_i^{(t)}\right)\_\min \\ c_2 = \left[F\left(x_i^{(t)}\right)\_\max + 0.5\right] \\ c_1 = \left[\varepsilon * F\left(x_i^{(t)}\right)\_\min + 0.5\right] \end{cases} \tag{6}$$

In formula (4), (5) and (6), ε = 1,2,3,4, [ ] indicates rounding operation. In the BRSPSO algorithm, in state II, the *i*-th particle has a positive impact on search, the social experience of the particle is reduced and its own experience plays a dominant position. While in state III, the representative *i*-th particle has a negative influence on search, and the social experience plays a leading role. In state I and state IV, particle velocity is mainly influenced by initial velocity, own experience and social experience.

During the iteration, the result of each iteration is saved to an array called *resultarray*, the algorithm will stop executing and find the optimization results in array *resultarray* when the number of iterations is satisfied with $T \geq T_{\max}$ or the *fitness score* = 0. Reassign the velocity and position of particles that are beyond the boundary D^N, when the particle position exceeds the boundary $\left(\left|p_{in}\right| \geq D, n = 1,2,...,N\right)$.

In the BRSPSO algorithm, in order to further improve the optimization ability of the algorithm, in addition to adjusting the inertia weight and the learning factor, the restart strategy is also introduced.

Huberman et al. [11] proposed in 1997 that restarting is a very economical strategy for complex computer problems. The restart strategy means that if the algorithm still fails to meet the requirements within a given time or iteration within a given number of times, then select the new random number seed restart algorithm and loop through the above steps until the algorithm terminates [12]. Guo et al. [13] can improve the performance by selecting better decision variables for the SAT problem by restarting the problem, the proposed strategy has significantly improved the key indexes such as memory occupation and the number of conflicts. And Chen et al. [12] put forward a efficient reboot strategy construction method, and analyzed the average performance and stability, and showed the value of its application in large-scale traveling salesman problem.

In the PSO algorithm, all particles start from the random solution. In the BRSPSO algorithm, the initial state of the particle is determined by two stochastic processes. The first random process determines the initial velocity of the particle $v_i\left(\xi_{\min} \leq v_i \leq \xi_{\max}\right)$. The second random process determines the initial position of the particle $p_i\left(\xi_{\min} \leq p_i \leq \xi_{\max}\right)$. If the value of v_i and p_i has never been seen, the program continues to execute. If it has occurred, the two random processes are repeated to reinitialize the particle until the initial velocity and initial position of the particle are not repeated. When $T < T_{\max}$, *fitness score* > 0, time of repetition $\theta > \theta_{\max}$, record the current iteration search results and save them to the array *resultarray*, then stop the operation of the algorithm and restart the strategy to re-execute the above steps. Finally, the algorithm optimization results can be obtained by comparing the minimum values in the array.

The steps of improving the BRSPSO algorithm are shown in Fig. 1.

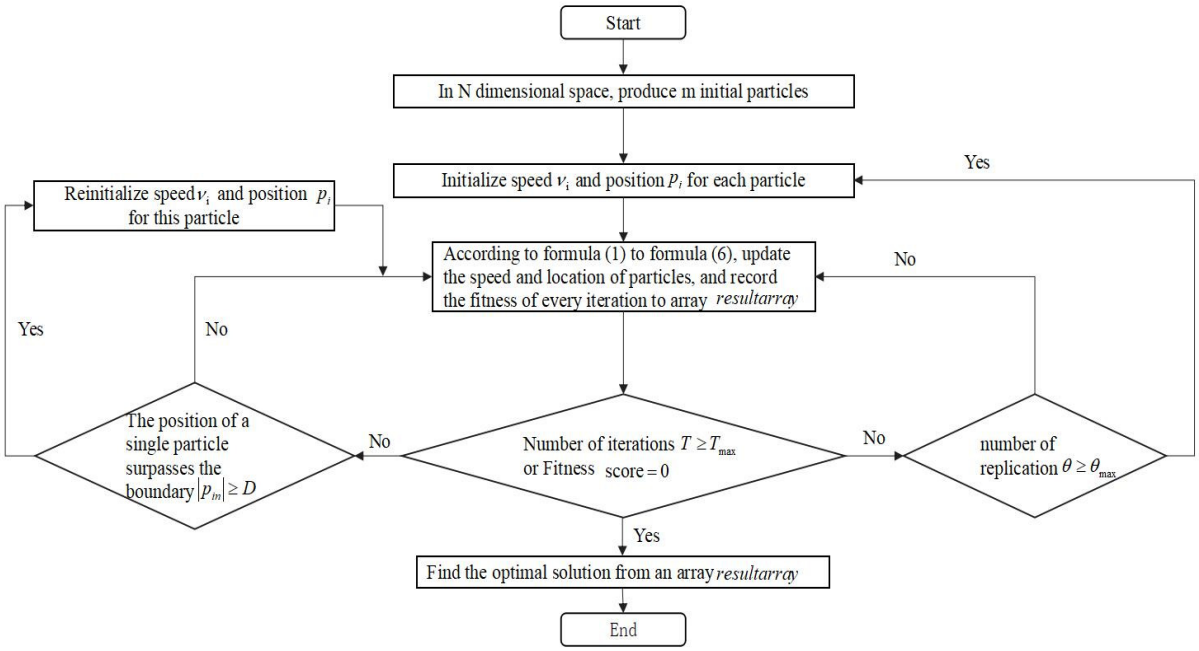

Fig. 1. The step diagram of the BRSPSO algorithm

4 Experimental Results and Analysis

4.1 Experimental Environment

This experiment uses Inter (R) Pentium (R) CPU G2020 @ 2.90GHz processor, 8.00GB RAM, 64 bit Windows7 operating system desktop computer. All algorithms in this paper are written in Python language with the tool pycharm.

4.2 Experimental Results and Analysis

In order to verify the effectiveness and feasibility of this improved particle optimization algorithm, this paper selects the data of Boston house price, which is typical and authoritative, as an application example of this method, and the optimization fitness function value of the algorithm is used as the evaluation criterion of the algorithm. As a case of forecasting the price of Boston house, set T_{max} = 500, θ_{max} = 10, ζ_{min} = 0, ζ_{max} = 1, D = 50, The Gauss kernel SVM is chosen to establish the regression model. The penalty coefficient C and the kernel coefficient *gamma* are the parameters to be optimized. In order to use 0 as the best result score, it uses the absolute value of the cross validation score minus 1 as the fitness function criterion of the improved particle swarm optimization algorithm. 14 characteristics of housing are taken as input variables, such as school district, supermarket, vegetable market and so on. Housing price is used as output variable to establish regression model. Under different values of ε, randomly run three times the algorithm proposed d in this paper and record the data, at the same time, compare with the results under the condition of traditional particle optimization algorithm in random three operation and no seeking optimization. The experimental data are as follows (the experimental results retained nine bits after the decimal point).

The SVM fitness score is 0.995 445410 when the particle swarm optimization algorithm is not used. Fig. 1 to Fig. 3 and Table 1. to Table 3. are the experimental results of three random running of SVM, PSO-SVM, and BRSPSO-SVM. The broken line in the line graph represents the fitness score of the different iterations number of algorithm. The closer the score is to 0, the better the algorithm's ability of searching is. It can be seen that the line graphs of SVM and PSO-SVM are basically a straight line, and the optimization results of the first two PSO-SVMs are higher than the fitness scores of the SVM algorithm, although the optimization results of the third operation are lower than The fitness score of the SVM algorithm, but the optimization of such a slightly better score is a random score, and it cannot

guarantee that each optimization can get better results, and These three PSO-SVMs all get the final optimization results at the number of iterations $T \leq 10$, and are higher than the optimization results of the improved algorithm, which shows that the algorithm converges too fast and has fallen into the local extremum.

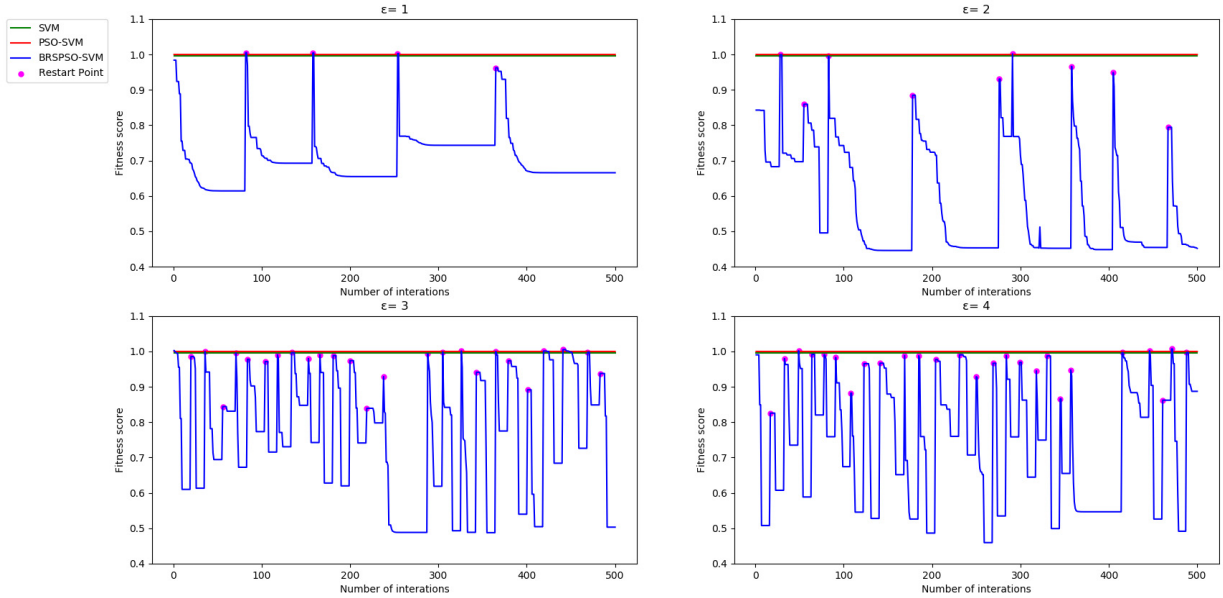

Fig. 2. *Fitness score* line chart of three kinds of algorithm at the first run

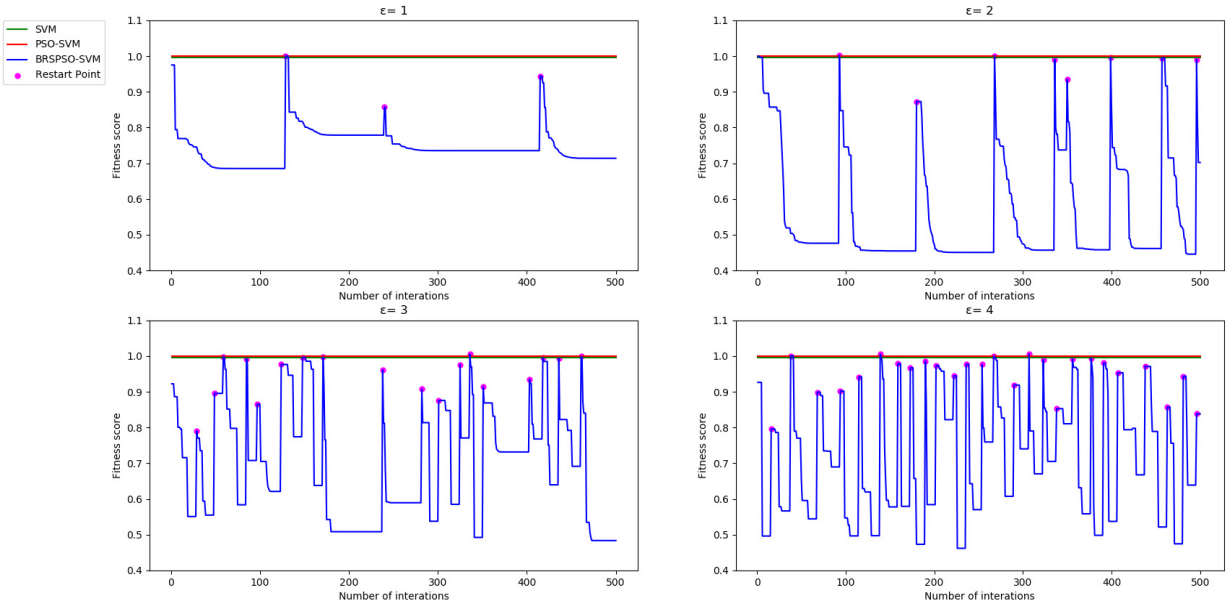

Fig. 3. *Fitness score* line chart of three kinds of algorithm at the second run

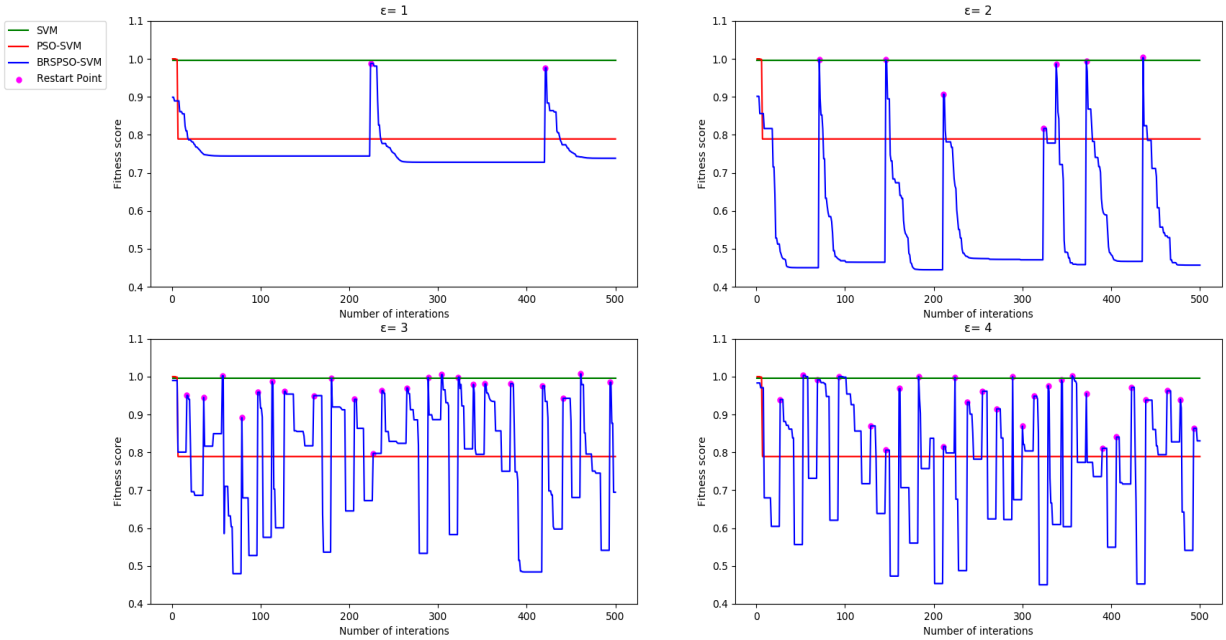

Fig. 4. *Fitness score* line chart of three kinds of algorithm at the third run

Table 1. *Fitness score* of three kinds of algorithm at the first run

Number of Iterations	Fitness Score					
	SVM	PSO-SVM	BRSPSO-SVM			
			ε=1	ε=2	ε=3	ε=4
1	0.995 445410	1.000 058436	0.984 017630	0.842 703521	1.002 564764	0.990 443279
2	-	1.000 004448	0.984 017630	0.842 703521	0.999 484299	0.990 443279
3	-	1.000 003012	0.984 017630	0.842 703521	0.996 654301	0.990 443279
4	-	1.000 000039	0.923 687203	0.842 703521	0.995 011766	0.990 443279
10	-	1.000 000039	0.754 713062	0.841 888278	0.609 934128	0.507 672281
50	-	1.000 000039	0.614 251415	0.696 830513	0.609 934128	0.966 437883
100	-	1.000 000039	0.715 211760	0.742 586300	0.609 934128	0.674 178237
200	-	1.000 000039	0.654 888257	0.723 670197	0.609 934128	0.486 290289
300	-	1.000 000039	0.743 247158	0.715 460035	0.609 934128	0.969 267156
400	-	1.000 000039	0.671 070837	0.448 077610	0.539 736872	0.546 521049
500	-	1.000 000039	0.665 538087	0.451 723886	0.503 113830	0.887 276568
Number of restarts	-	-	4	9	25	26
Optimization result	0.995 445410	1.000 000039	0.614 141469	0.445 700105	0.487 505842	0.459 350211

Table 2. *Fitness score* of three kinds of algorithm at the second run

Number of Iterations	Fitness Score					
	SVM	PSO-SVM	BRSPSO-SVM			
			ε=1	ε=2	ε=3	ε=4
1	0.995 445410	1.000 001007	0.975 150749	0.999 446871	0.922 407156	0.926 355310
2	-	1.000 001007	0.975 150749	0.999 446871	0.922 407156	0.926 355310
3	-	1.000 001007	0.975 150749	0.999 446871	0.922 407156	0.926 355310
4	-	1.000 001007	0.975 150749	0.995 964571	0.886 374483	0.926 355310
10	-	1.000 001007	0.769 123063	0.895 952608	0.800 595946	0.496 192411
50	-	1.000 001007	0.687 301717	0.478 878793	0.895 583518	0.653 988008
100	-	1.000 001007	0.685 362169	0.745 647201	0.865 175559	0.546 876782
200	-	1.000 001007	0.778 699309	0.473 375772	0.508 211723	0.584 096748
300	-	1.000 001007	0.735 373702	0.474 047797	0.537 613594	0.740 218748
400	-	1.000 001007	0.735 363336	0.856 117609	0.731 551243	0.537 208219
500	-	1.000 001007	0.713 874968	0.702 085397	0.483 586683	0.838 270903
Number of restarts	-	-	3	8	18	26
Optimization result	0.995 445410	1.000 001007	0.685 362140	0.445 473300	0.445 700105	0.461 930627

Table 3. *Fitness score* of three kinds of algorithm at the third run

Number of Iterations	Fitness Score					
	SVM	PSO-SVM	BRSPSO-SVM			
			ε=1	ε=2	ε=3	ε=4
1	0.995 445410	1.000 033203	0.899 001796	0.901 372458	0.989 937583	0.983 275814
2	-	1.000 000000	0.899 001796	0.901 372458	0.989 937583	0.983 275814
3	-	0.999 995821	0.889 359242	0.901 372458	0.989 937583	0.983 275814
4	-	0.998 658325	0.889 359242	0.856 030452	0.989 937583	0.983 275814
10	-	0.789 095303	0.860 914270	0.816 483508	0.800 595946	0.679 816803
50	-	0.789 095303	0.744 521430	0.450 039396	0.849 424407	0.556 464744
100	-	0.789 095303	0.744 144782	0.468 324644	0.916 796118	0.998 563822
200	-	0.789 095303	0.744 144759	0.444 639439	0.645 271846	0.837 186360
300	-	0.789 095303	0.727 814987	0.470 860246	0.886 524740	0.869 196401
400	-	0.789 095303	0.727 814883	0.484 082482	0.484 061269	0.549 383810
500	-	0.789 095303	0.738 259173	0.456 777768	0.694 581591	0.830 442548
Number of restarts	-	-	2	7	23	27
Optimization result	0.995 445410	0.789 095303	0.727 814883	0.444637781	0.479 590392	0.450 220880

The blue line in Fig. 1 to Fig. 3 is the result of the BRSPSO-SVM experiment at the time of three random runs when $\varepsilon = 1, 2, 3, 4$. The dot of line chart represents algorithm restarting point which refers to repetition times θ of optimization results $\theta \leq 10$. It can be seen from the experimental results that the higher value of and the more restart times, when $\varepsilon = 1$, the restart times of the algorithm are the least, but the optimization results are higher than when $\varepsilon = 2$, indicating that the exploration ability is not enough at this time. When $\varepsilon = 3,4$, though the number of restart times was increased, it may be the wrong target area at this time, which results in the optimization result which is still higher than the result of situation when $\varepsilon = 2$. Even so, the optimization results of BRSPSO-SVM algorithm are lower than the optimization results of PSO-SVM algorithm and SVM. When $\varepsilon = 2$, the third time running BRSPSO-SVM is 55.332% lower than that of SVM algorithm, which is better than PSO- The SVM experiment results were reduced by a maximum of 55.536%. In the BRSPSO algorithm, as the number of iterations increases, the algorithm keeps restarting, and the search range of the algorithm is gradually expanded, which avoids the algorithm falling into local extremum. This shows that the BRSPSO algorithm can find the domain of global optimal value more effectively.

5 Conclusion

In order to solve the problems of premature convergence and local extremum of traditional PSO algorithm, this paper proposes a BRSPSO method, update speed and position iteration formula on the basis of the interaction between particles by introducing a restart strategy. In this paper, the BRSPSO algorithm is used to optimize the parameters of SVM, compared with the results of optimized by traditional PSO algorithm and the results of SVM without PSO, the simulation results show that,

(1) PSO-SVM obtains the final optimization result when the number of iterations is less than 10, which indicates that the PSO-SVM algorithm converges too fast and the search range is limited. The BRSPSO algorithm further expands the particle search range and avoid falling into local extremum by giving the particles different initial states through the restart strategy.

(2) The results of BRSPSO algorithm are lower than the values of PSO-SVM and SVM. BRSPSO algorithm enhances the influence of excellent particles and the particle search capability according to the influence between particles, and introduce particle boundary values to prevent severe oscillations in the late stage of the particle.

(3) The optimization ability of BRSPSO algorithm is stronger than the traditional PSO algorithm's. However, the BRSPSO algorithm increases the randomness of the particles, which causes the result of optimization is stochastic and the optimal results are related to the number of iterations. Generally, the larger the number of iterations, the larger the particle search range, and the better the optimization result.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. the IEEE International Conference on Neural Networks, 1995.

[2] Y.G He, W.G. Zhu, Y.T. Zhou, M. R. Rong, An analog circuit diagnosis method based on particle swarm optimization Algorithm, Transactions of China Electrotechnical Society 25(6)(2010) 163-171.

[3] H. Wang, H.B. Ouyang, L.Q. Gao, An improved global particle swarm optimization, Control and Decision 31(7)(2016) 1161-1168.

[4] X.S. Jiang, J. Ren, M.M. Gu, Multi-dimensional descending chaotic inertia weight based PSO and its application, Chinese Journal of Scientific Instrument 36(6)(2015) 1333-1341.

[5] X.L. Zhu, J. Ling, B. Wang, J.H. Hao, Y.H. Ding, Soft-sensing modeling of marine protease fermentation process based on improved PSO-RBFNN, Journal of Chemical Industry and Engineering(China) 69(3)(2018) 1221-1227.

[6] Z.Y. Luo, L. Han, C.Y. Luo, Z. Jin, C. Yuan, H. C. Su, genetic-particle swarm memetic algorithm and application in optimization design of dual stator-winding induction generator for wind turbine, Acta Energiae Solaris Sinica 38(4)(2017) 928-937.

[7] J.J. Liu, J.L. Yang, B. Sun, X.T. Zhang, H. Meng, Application of smell and taste information fusion technology in classification of beer based on particle swarm optimization, Transactions of the Chinese Society of Agricultural Machinery 47(10)(2016) 244-249.

[8] Y.Z. Wang, Y.J. Lei, Improved adaptive particle swarm optimization algorithm based on best fitness evaluation, Systems Engineering and Electronics 30(12)(2008) 2497-2501.

[9] X.P. Xu, F.C. Qian, F. Wang, Novel method of system identification based on modified particle swarm optimization algorithm, Systems Engineering and Electronics 30(11)(2008) 2231-2236.

[10] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proc. the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), 1999.

[11]B.A. Huberman, R.M. Lukose, T. Hogg, An economics approach to hard computational problems, Science 275(5296)(1997) 51-54.

[12] G.L. Chen, X. Xie, Y. Xu, J. Gu, Designing restart strategy for randomized algorithms and its application in solving the TSP, Chinese Journal of Computers 25(5)(2002) 514-519.

[13] Y. Guo, C.S. Zhang, B. Zhang, An dynamic restart strategy for solving SAT problem, Journal of Northeastern University(Natural Science) 35(7)(2014) 0-0.