参赛队员姓名：唐祯明

中学：四川省成都市树德中学（光华校区）

省份：四川

国家/地区：中国

指导教师姓名：陈雪峰

指导教师单位：四川省成都市树德中学

论文题目：An Improved Multidimensional Particle Swarm Algorithm and its Application in Nucleic Acid Test(NAT) Arrangement

An Multidimensional Improved Particle Swarm Algorithm and its Application in Nucleic Acid Test(NAT) Arrangement

Tang Zhenming

Abstract Particle Swarm Algorithm is an available global search algorithm influenced by the behavior of birds seeking food. It shows rapid speed in optimization and optima searching. However, the limitation of the standard form is obvious as well— because of the small inertia factor and personal learning factor which were set at very beginning, the particles may cannot search very widely and easily fall into local optima. The solution to this problem is the main purpose of this article. This paper presents improvement of PSO by changing the inertia factor, personal learning factor and social learning factor through the searching process and combining Simulated Annealing Algorithm. Then it presents an application of the improved PSO algorithm—— Optimization of the scheduling of nucleic acid specimen collection vehicles in the context of the COVID-19 epidemic.

Keywords: Particle Swarm Algorithm, Descent Method, Simulated Annealing Algorithm, Cauchy Mutation, ANOVA, Convergence Analysis, Optimization Solution

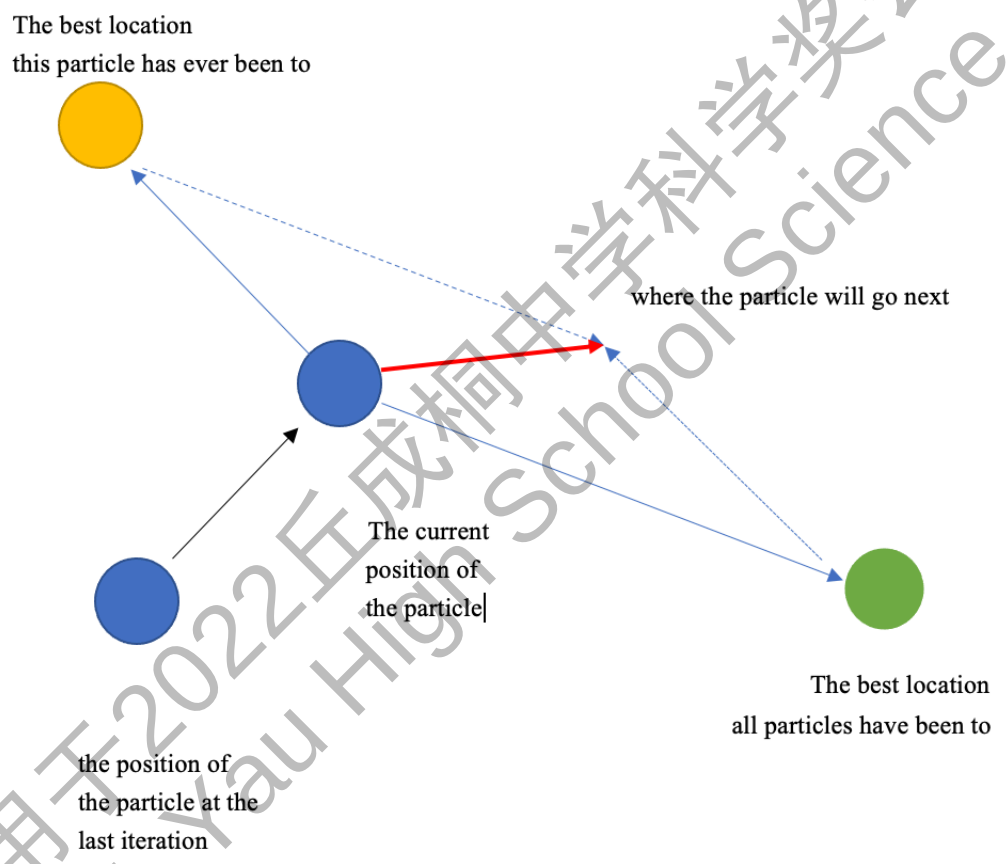
I. Introduction

In 1995, American social psychologist J. Kennedy and electrical engineer R. Eberhart firstly proposed Particle Swarm Algorithm.[1] The Particle Swarm Optimization (PSO) is an evolutionary algorithm which has shown great performance in optimization searching[2] . Unlike other evolutionary algorithm, it uses fewer parameters and easy to achieve the goal, which specially motivated the simulation of social behavior that enabled it to get faster convergence velocity to the target point (game best point). Due to its advantages, the PSO is increasingly used in neural network training[3], pattern classification[4] and other fields.

Like other nature-based algorithm such as Ant colony Optimization[5], it starts with a limited population index and randomly generated solutions called particles and each particle has a certain position and velocity calculated by Eqs(1-1) and (1-2) in order to seek the game best position[6]. In the earliest version of PSO which J. Kennedy and R. Eberhart proposed in 1995, the Eqs(1-1) and (1-2) were presented as:

$$v_i^d = v_i^d + c_1 rand_1^d\left(pbest_i^d - x_i^d\right) + c_2 rand_2^d\left(gbest^d - x_i^d\right) \quad (1\text{-}1)$$

$$x_i^d = x_i^d + v_i^d \quad (1\text{-}2)$$



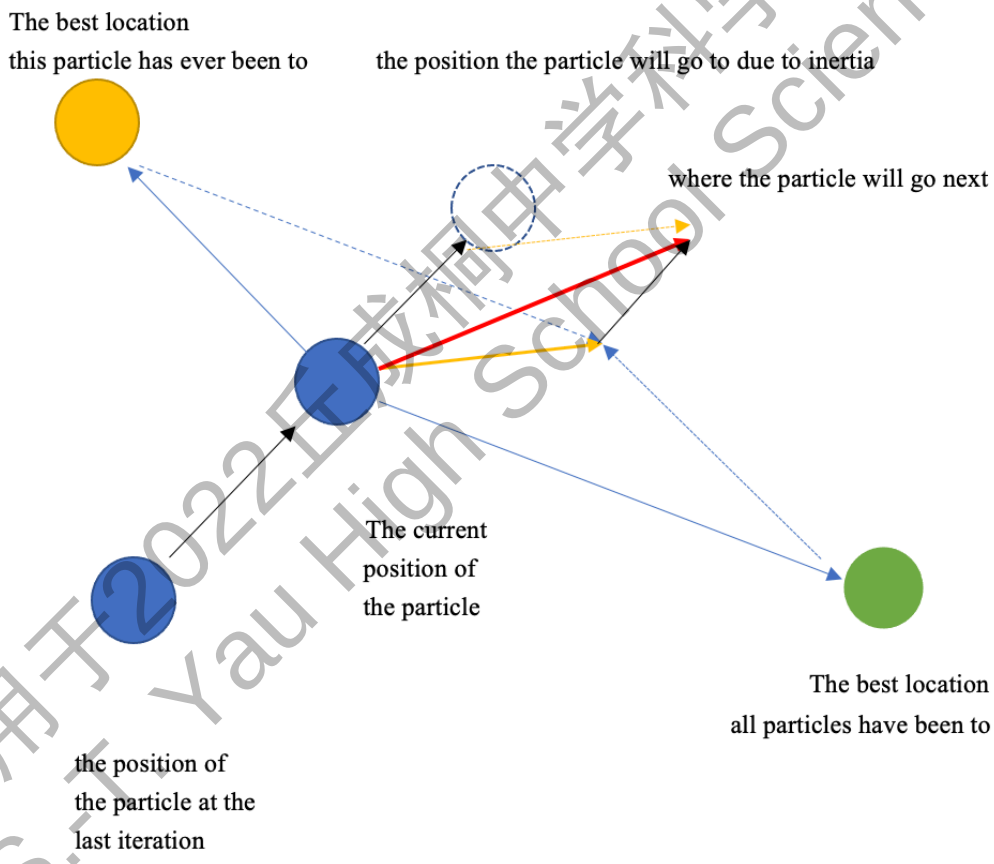
Figure(1-1) Particle swarm algorithm illustration

where all superscript d means the dth dimension, v_i^d and x_i^d are the velocity and position of the ith particle; c_1 and c_2 are the personal learning factor and social learning factor; $pbest_i^d$ and $gbest^d$ are the best position of the point that the ith particle has found so far and the position of the point that all particles has found so far respectively; $rand_1^d$ and $rand_2^d$ are two random numbers above 0 and this paper will present their improvement way later.

In the Eq(1-1), we can find that the inertia factor ω has not been proposed yet. The earliest one is able to find the optima successfully without doubt, but it may probably fall into local optima though because the Eq(1-1) lessen the particles' "self-conscious" and are more likely to iterate to the game best point at the beginning while the real game best point probably does not appear towards that direction.

Therefore, Y. Shi and R. Eberhart proposed inertia factor ω in 1997.[7] They improved Eq(1-1) to Eq(1-1′):

$$v_i^d = \omega v_i^d + c_1 rand_1^d\left(pbest_i^d - x_i^d\right) + c_2 rand_2^d\left(gbest^d - x_i^d\right) \quad (1-1')$$



Figure(1-1') Particle swarm algorithm illustration (with inertia factor ω)

where ω is the inertia factor and usually a constant number. Compared with Eq(1-1), it avoids the circumstance above to some extent, but according to adequate simulation, the possibility of the particles fall into local optima is located at a relatively high level still. Its essential reason is that the particles cannot search as much area as possible if ω is too small or convergence speed will be too low while ω is much higher.

Nowadays, PSO also has good process in some practical application fields such as Power Systems[8] and Synthetic Molecules[9]. First, the researchers of Fuji Electric Power Co., Ltd. turn the Reactive Power and Voltage Control (RPVC) into function optimization problem using PSO to solve. Second, neural network training combining PSO with Back Propagation (BP) has been used to simulate the charging of electric vehicle fuel cell stacks. Its simulation accuracy is much higher than which Eberhart, Simpson and Dobbins proposed in 1996, according to the experiment data. Last but not least, PSO has been used by a biological company in the United States to optimize the combination of various biochemical components, and then artificially synthesize microorganisms.[10] It can be seen that PSO and its improved ones has been used in many disciplines other than computer science and mathematics, making great contribution to solving problems in practical scenarios. Additionally, PSO is also very helpful for us to solve life problems: Li Ning[11] applies PSO to vehicle path planning problem, Qi Xuemei[12] used PSO to solve the flow shop scheduling problem, etc. Similarly, we can apply PSO to the NAT arrangement in order to improve the speed and efficiency of nucleic acid detection to help fight the epidemic.

To better standard PSO, **Section II** shows the methodology which motivates a variety of decreasing strategies of ω and c_1 to improve its search space and convergence speed and combine it with Simulated Annealing Algorithm (SAA) and Cauchy mutation. In **section III**, variance analysis and convergence analysis will be done on the parameters and the improved algorithm which this paper will bring up later respectively in order to analyze theoretically. In **Section IV** the improved algorithm will be used in order to optimize the scheduling of nucleic acid specimen collection vehicles in Chenghua District, Chengdu, China against the global backdrop of COVID-19.

II. Methodology

This section improves the standard PSO by adding decreasing strategy of parameters ω and c_1, then combining with SAA and Cauchy mutation in order to speed up convergence and prevent struggling in a local optima. The first part of this section there will discuss five kinds of decreasing methodology and pick out one methods with the best overall performance and enter the second part of the discussion on the SAA and Cauchy mutation.

2-1 Descent Methods

The goal of the decreasing strategy is to make particles search more area at the early stage of the searching process and increase the convergence velocity at the late stage. Generally, there are two main kinds of descent method: linear ones and nonlinear ones. Now there proposes five kinds of methods. In the following formula, ω_0, ω_i, ω_k are the index of inertia factor at the start, after the *i*th iteration and at the end respectively (The expression of c_1 is the same); $\varphi = c_1 + c_2$.

First, there is the only linear one among them[13]:

Method 1:

$$\omega_i = \omega_0 - \frac{\omega_0 - \omega_k}{k} i$$
$$c_{1i} = c_{10} - \frac{c_{10} - c_{1k}}{k} i$$

The connection of ω_0 and ω_k, c_0 and c_k is a linear function with slope equals to $-\frac{\omega_0-\omega_k}{k}$ and $-\frac{c_{10}-c_{1k}}{k}$ respectively. We refer to this method **Lin** in the following discussion.

Then there are four kinds of nonlinear kinds:

Method 2:

$$\omega_i = -(\omega_0 - \omega_k)\left(\frac{i}{100}\right)^2 + \omega_0$$

$$c_{1i} = -(c_{10} - c_{1k})\left(\frac{i}{100}\right)^2 + c_{10}$$

Method 3:

$$\omega_i = (\omega_0 - \omega_k)\left(\frac{i}{100}\right)^2 + (\omega_k - \omega_0)\left(\frac{2i}{k}\right) + \omega_0$$

$$c_{1i} = (c_{10} - c_{1k})\left(\frac{i}{100}\right)^2 + (c_{1k} - c_{10})\left(\frac{2i}{k}\right) + c_{10}$$

these two functions[13] are both quadratic while the first one is convex and the other one is concave. The convex one and the concave one are named **Convex** and **Concave** respectively.

Method 4:

$$\omega_i = \omega_0 - \frac{\omega_0 - \omega_k}{k - i}(i \le k - 1)$$

$$c_{1i} = c_{10} - \frac{c_{10} - c_{1k}}{k - i}$$

whose figure likes an inversed inverse proportional function and we will call it **I-Inverse**.

Method 5:

$$\omega = \frac{2}{\left|2 - \varphi - \frac{\varphi^2 - 4\varphi}{2}\right|}$$

The above-mentioned PSO with contraction factor was proposed in order to improve the searching ability of the algorithm. [14] Because the method was proposed by Clerc and Kennedy, there is no harm in calling it **CK**.

Figure(2-1-1) shows the curves of decreasing strategy function with the increase of iteration in five kinds of methods.

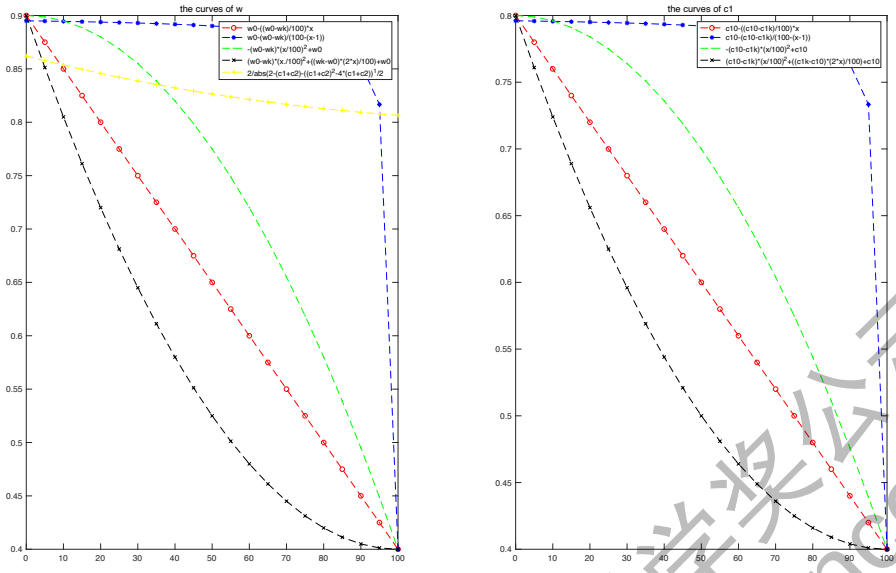


Figure (2-1-1) the curves of decreasing strategy function with the increase of iteration in five kinds of methods

2-2 Simulated Annealing Algorithm

An obvious difference between PSO and SAA is that SAA will probably accept a point that is not the actual personal best point so far as a new personal best point. This allows points to get rid of the local optima then search wider area.

As SAA has probability to accept a point that is not a personal best point, there will be a function to express this probability. In 1953, M. Carlo proposed Metropolis criterion shown as(suppose the goal is to search the minimum)

$$P = \begin{cases} e^{\frac{\Delta f}{kT}}, & f(x_1) > f(x_0) \\ 1, & f(x_1) \le f(x_0) \end{cases} \tag{2-2-1}$$

$$T_i = T_{i-1} \times dT \tag{2-2-2}$$

where Δf is $-|f(x_1) > f(x_0)|$, a negative real number; T and dT are the current annealing temperature and the decreasing rate of T respectively and k is Boltzmann parameter, which equals to 1 in the algorithm.

In SAA, T and dT are the parameters that mainly affect the performance of the algorithm. If the value T is large and dT is small, the particles' moving range will be wide and if not it will be narrow.

2-3 Cauchy Mutation

Through previous dynamic image and theoretical results[15] that the particles in PSO will oscillate between their previous personal best point and the game best point so far before it converges. If one can apply a mutation to some particles when they enter an optima, it will extend searching area thus get rid of the local optima and increase success rate. This can be accomplished by having Cauchy Mutation on the particles in every generation.[16] The one-dimensional Cauchy density function centered at the origin is defined by:

$$p(x) = \frac{1}{\pi}\frac{t}{t^2+x^2} \quad (x \in (-\infty, +\infty)) \tag{2-3-1}$$

where $t > 0$ and it equals to 1 in this paper. The Cauchy distributed function is:

$$F(x) = \int_{-\infty}^{x} p(x) = \frac{1}{\pi}\arctan(x) + \frac{1}{2} \tag{2-3-2}$$

Then with Eq(4-2), it is able to apply Cauchy mutation to the present game best point according to Eq(4-3)

$$x_{i(t+1)}^{d} = x_{it}^{d} \times (n\eta \cdot F(rand(x)) + 1) \tag{2-3-3}$$

and

$$n = \frac{Iteration\ times - Current\ interation\ times}{Iteration\ times}$$

$$\eta > 0$$

where x_{it}^{d} and is the location of the *i*th particle after *t* iteration times at the *d*th dimension; η is a positive parameter to increase or decrease the particles' jump length depending on the function; n is a parameter that modify convergence speed of early stage and lately stage; $F(x)$ is Cauchy distributed function and $rand(x)$ is a random number belongs to $(-\infty, +\infty)$.[17][18]

The pseudocode below shows the pseudocode of the entire process of the improved PSO of this paper, which combine decreasing strategy, SAA and Cauchy mutation.

Begin

$N = population\ size$
$d = dimension$
$iter = current\ iteration\ time$
$ger = iteration\ times$
$limit[\ \ ] = location\ restrictions$
$vlimit[\ \ ] = velocity\ restrictions$
$xm = personal\ historical\ best\ position\ (pbest)$
$fxm = personal\ historical\ best\ fitness$
$ym = game\ historical\ best\ position\ (gbest)$
$fym = game\ historical\ best\ fitness$
$T = annealing\ temperature$
$dT = annealing\ temperature\ reduction$

$\boldsymbol{while}(iter < ger)$
$Update\ the\ value\ of\ \omega\ and\ c_1\ according\ to\ method\ \boldsymbol{CK}$
$Update\ the\ velocity\ a\ of\ each\ particle\ according\ to\ Eq(1-1')$
$\boldsymbol{if}\ (velocity > vlimit\ or\ velocity < vlimit)$
$velocity = vlimit$
$\boldsymbol{if}\ end$
$Update\ the\ positions\ of\ each\ particle\ according\ to\ Eq(1-2)$
$\boldsymbol{if}\ (position > limit\ or\ position < limit)$
$position = limit$
$\boldsymbol{if}\ end$
$Calculate\ the\ fitness\ value$
$\boldsymbol{for}\ i = 1{:}N$
$Update\ xm\ and\ fxm\ according\ to\ SAA$
$\boldsymbol{for}\ end$
$Update\ ym\ and\ fym\ if\ needed$
$Calculate\ Cauchy\ mutation\ index$
$Add\ Cauchy\ mutation\ to\ the\ particles\ to\ Eq(4-3)$
$\boldsymbol{if}\ (position > limit\ or\ position < limit)$
$position = limit$
$\boldsymbol{if}\ end$
$Update\ fxm\ if\ the\ paticle's fitness\ value < fym. Otherwise, Update\ fxm\ according\ to\ SAA$
$Update\ fym\ if\ needed$
$T = T \times dT$
$iter = iter + 1$
$\boldsymbol{while\ end}$
End

From chapter **2-1** to **2-3**, it shows the entire improved algorithm process of PSO that this paper intend to propose. Because it includes descent method of ω and c_1, SAA and Cauchy mutation, the improved PSO is named **DSCPSO** in coming discussions.

2-4 Illustrations

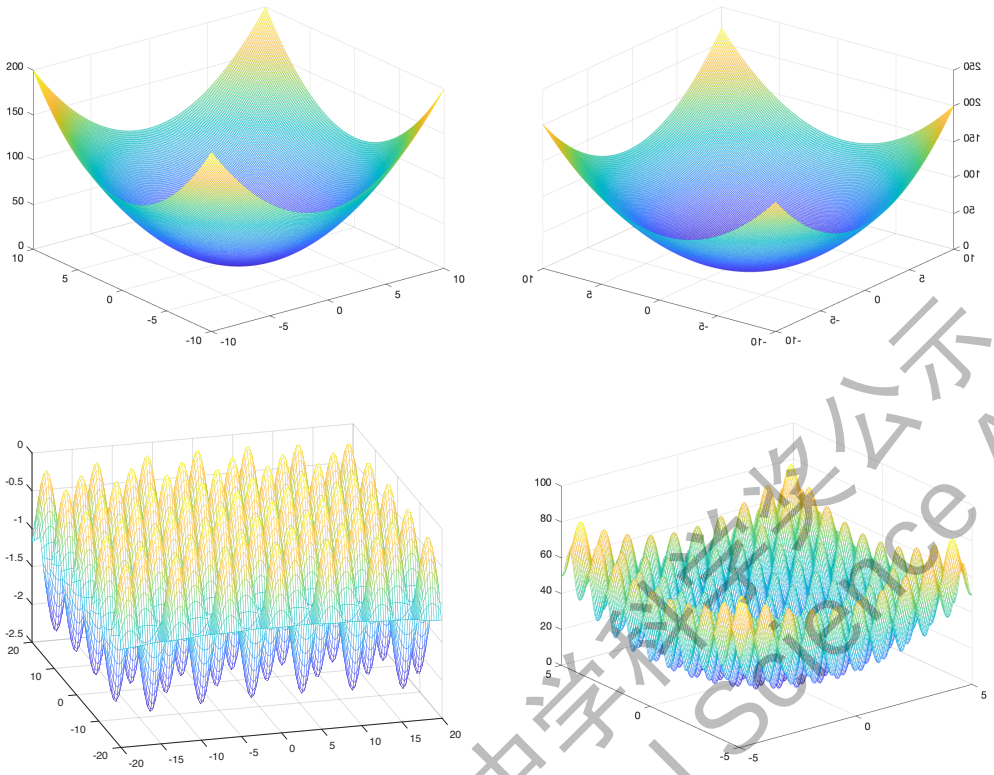
2-4-1 Descent Method

Set $c_{10} = c_2 = 0.8, \omega_0 = 0.9$ and $k = 100$; for **Lin** to **Concave** $c_{1k} = \omega_k = 0.4$, and the decreasing strategy of ω and c_1 in **CK** is the one in **LIN** by default.

In order to examine the performances of five methods, four kinds of classic test functions shown in Table(2-4-1-1) are used and their figures in 2-dimension are shown in figure(2-4-1-2)

FUNCTION NAME	***FUNCTION***	***DIMENSION***	***SEARCHING SPACE***	***MAX SPEED***	***BREAK CONDITION***
SPHERE	$f1 = \sum_{i=1}^{d} x_i^2$	10	$(-10,10)^d$	0.5	0.001
STEP	$f2 = \sum_{i=1}^{d} \lvert x_i + 0.5 \rvert^2$	10	$(-10,10)^d$	0.5	0.001
GRIEWANK	$f3 = \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	10	$(-10,10)^d$	0.5	0.1
RASTRIGRIN	$f4 = \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i) + 10]$	10	$(-10,10)^d$	0.5	/

Table(2-4-1-1) the four classic test functions for convergence speed examination



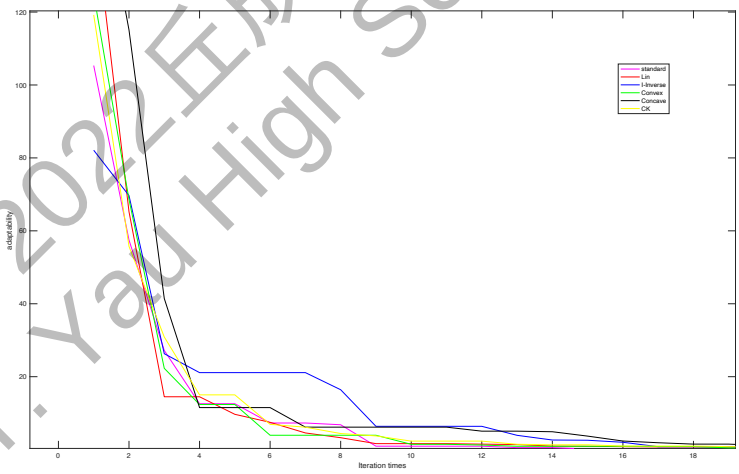
Figure(2-4-1-2) the figures of the test function(From top left to bottom right: *Sphere*, *Step*, *Griewank* and *Rastrigrin*)

whose minimum are all 0 when all the elements equal to 0. And it is not hard to find that *Sphere* and *Step* are unimodal functions and *Griewank* and *Rastrigrin* are multi - 6modal functions. Set $N = 100$, $velocity\ limit \in [-10,10]$ (if the velocity index is above 10 or below -10, then the index equals to 10 or -10 compulsively). There will test the convergence speed of **LIN** to **CK**(including the standard one), the test standard is that total the iteration times while the result is below to 0.001 for *Sphere* and *Step* and 0.1 for *Griewank* (because the span of *Griewank* on z-axis is much smaller than *Sphere* and *Step*, therefore it is harder for *Griewank* to converge to a enough accurate small value) and for 1000 times and calculate the average of each method(if one result cannot converge below to 0.001 or 0.1 after 1000 iteration times, add 1000 to the total). Specially, because *Rastrigrin* is quite difficult to converge to a value that small enough, therefore there calculates the average of the final converged valve(after 200 iteration times) of each Method.

Table(2-4-1-3) to Table(2-4-1-6) show the test result of each method for each test function and Figure(2-4-1-3-1) to Figure(2-4-1-5-1) show the velocity of adaptability decreasing curves of each method.(Note: Due to the particularity of *Rastrigrin*, we can judge its convergence performance by variance.)

ALGORITHM	STANDARD	LIN	I-INVERSE	CONVEX	CONCAVE	CK
TEST TIME	1000	1000	1000	1000	1000	1000
AVERAGE ITERATION TIMES	117.33	95.39	337.45	94.74	417.11	74.28
VARIANCE WHEN CONVERGE BELOW TO 0.001	7.6328×10^{-7}	8.7094×10^{-5}	2.502×10^{-4}	5.7115×10^{-7}	1.2063×10^{-11}	4.2056×10^{-5}
VARIANCE AFTER 1000 ITERATION TIMES	4.3593×10^{-40}	1.0285×10^{-52}	8.6907×10^{-42}	4.1195×10^{-58}	1.3333×10^{-46}	8.4568×10^{-55}

Table(2-4-1-3) the running result of the average iteration times of each Method for *Sphere*



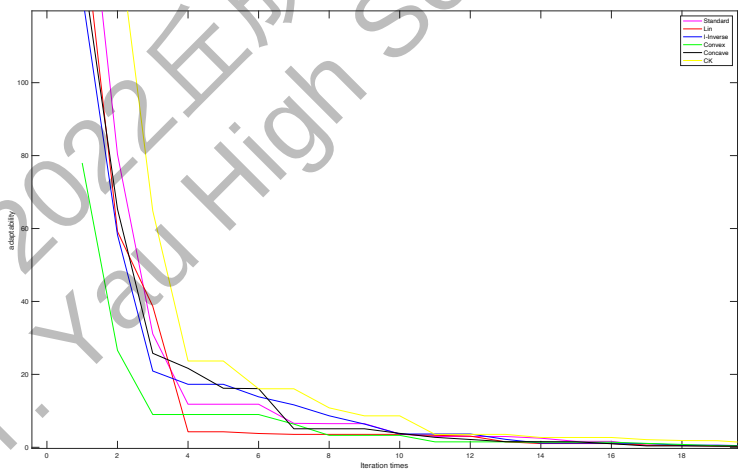
Figure(2-4-1-3-1), the velocity of adaptability decreasing curves of each method within iteration times[0,20] for *Sphere*

According to the table(2-4-3-1) and the Figure(2-4-3-1-1), **CK** has the fastest convergence speed, an average of about 74 iterations converged to the relatively ideal results and **Convex** and **Lin** are the second and the third, 95 times and 96 times

respectively. For the decreasing velocity, the standard one drops fastest; **Lin**, **Convex**, **CK** drop slower than the standard one and **I-Inverse** as well as **Concave** drops slowest. Moreover, the convergence accuracy of **I-Inverse** and **Concave** are not ideal, so that their average iteration times are increased.

ALGORITHM	STANDARD	LIN	I-INVERSE	CONVEX	CONCAVE	CK
TEST TIME	1000	1000	1000	1000	1000	1000
AVERAGE ITERATION TIMES	122.38	90.69	288.11	90.96	449.81	68.94
VARIANCE WHEN CONVERGE BELOW TO 0.001	5.4008×10^{-4}	1.7696×10^{-5}	9.3524×10^{-4}	2.1864×10^{-8}	3.0457×10^{-4}	9.3929×10^{-4}
VARIANCE AFTER 1000 ITERATION TIMES	1.935×10^{-39}	3.5156×10^{-54}	1.7889×10^{-41}	5.0821×10^{-58}	77.9380×10^{-49}	8.9944×10^{-54}

Table(2-4-1-4) the running result of the average iteration times of each Method for *Step*



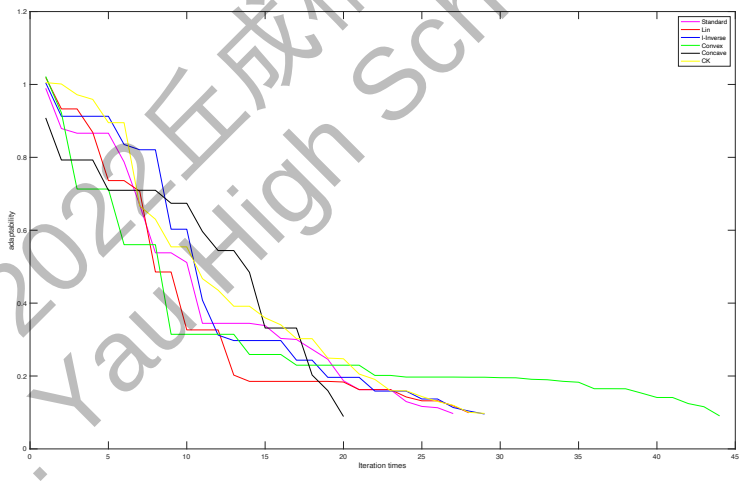
Figure(2-4-1-4-1), the velocity of adaptability decreasing curves of each method within iteration times [0,20] for *Step*

According to the table(2-4-1-4) and figure(2-4-1-4-1), **CK** has the fastest convergence speed, an average of about 69 iterations converged to the relatively ideal results and **Convex** and **Lin** are the second and the third, about 90 times. For the decreasing velocity, **Lin** and **Convex** drop fastest at the early stage while **CK** drops

slowest in all 6 methods. However, the convergence effects of other methods are not ideal, so that their average iteration times are increased.

ALGORITHM	STANDARD	LIN	I-INVERSE	CONVEX	CONCAVE	CK
TEST TIME	1000	1000	1000	1000	1000	1000
AVERAGE ITERATION TIMES	137.76	166.42	100.97	150.06	137.17	95.94
VARIANCE WHEN CONVERGE BELOW TO 0.1	9.988×10^{-2}	8.8888×10^{-2}	9.7919×10^{-2}	7.5295×10^{-2}	7.9835×10^{-2}	7.1895×10^{-2}
VARIANCE AFTER 1000 ITERATION TIMES	9.2965×10^{-33}	7.0393×10^{-33}	7.0187×10^{-33}	7.0562×10^{-33}	9.8642×10^{-33}	9.9803×10^{-34}

Table(2-4-1-5) the running result of the average iteration times of each Method for *Griewank*



Figure(2-4-1-5-1) he velocity of adaptability decreasing curves of each method within iteration times for *Griewank*

According to the table(2-4-1-5) and figure(2-4-1-5-1), **CK** has the fastest convergence speed, an average of about 96 iterations converged to the relatively ideal results and **I-Inverse** and **Concave** are the second and the third, 101 times and 137 times respectively. For the decreasing velocity, **Convex** drops very fast at the early stage but become much slower at the middle and late stage, thus slow down its

convergence speed. **Concave** drops slowest at the early stage while drops fastest at the late stage so that it has a faster convergence speed.

ALGORITHM	STANDARD	LIN	I-INVERSE	CONVEX	CONCAVE	CK
TEST TIME	1000	1000	1000	1000	1000	1000
AVERAGE FINAL CONVERGED VALUE	20.9983	23.1234	20.6139	22.3150	23.7780	22.3574
VARIANCE	167.97	8.837×10^{-28}	49.2879	2.0735×10^{-22}	1.3947×10^{-27}	8.5032×10^{-10}

Table(2-4-1-6) the running result of the average iteration times of each Method for *Rastrigrin*

According to table(2-4-1-6), **I-Inverse** and the standard one plays best which could converge to about 20.61 and 21 after 200 iteration times respectively, while **Concave** and **Lin** plays worst which could only converge to about 24 and 23 respectively. Note that the variance of **I-Inverse** and the standard one are quite big while **Concave** and **Lin**'s are quite small, which indicates that **Concave** and **Lin** are easy to fall into a local optima. **I-Inverse** and the standard one have better global searching ability that not easily disturbed by local optima.

In conclusion, **CK** runs best in unimodal functions as well as multimodal functions and **Convex** and **Lin** follow it, **CK** and **I-Inverse** run well in multimodal functions especially in *Griewank.* As for the variances, **Convex** and **CK** have better convergence speed in unimodal functions but in multimodal functions the differences between 4 test functions are not very significant, while **Convex** and **CK** are a little bit better, comparing with other 4 methods.

Take all the test data into consideration, this paper will take **CK** into coming research.

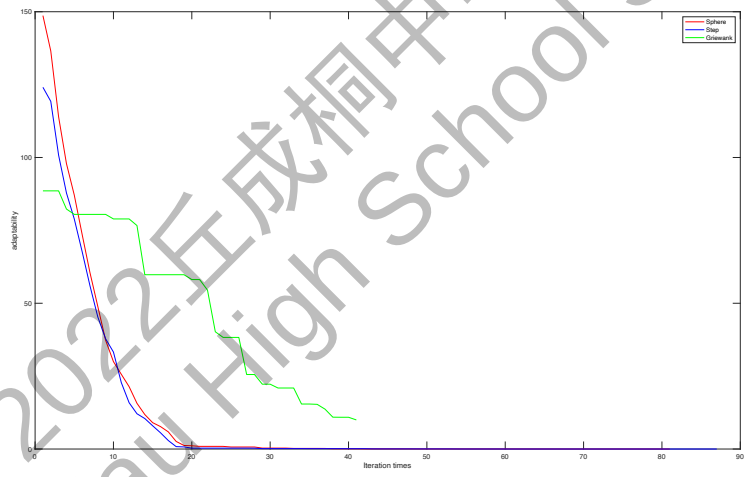
2-4-2 Simulated Annealing Algorithm

Set $T = 2000$ and $dT = 0.98$, the location and velocity of the particles abide by Eq(1-2) and Eq(1-1') as well. Table(2-10) lists the average convergence speed of

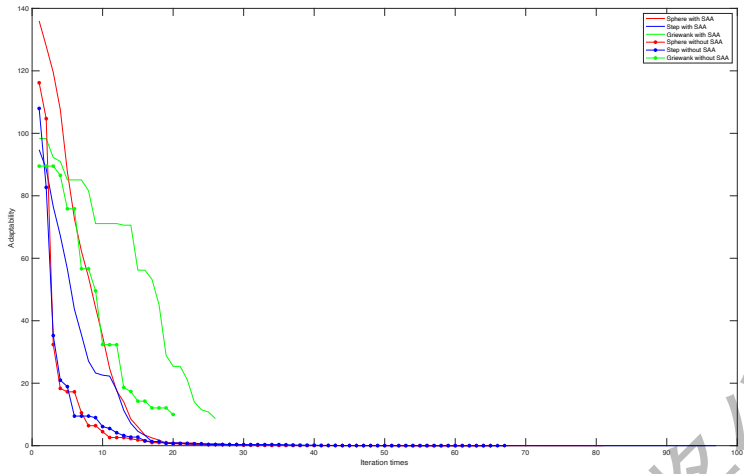
Convex and **CK** after combined with SAA. Figure(2-9) and Figure(2-10) show the running results of horizontal and vertical comparisons

CK	***SPHERE***	***STEP***	***GRIEWANK***	***RASTRIGRIN***
AVERAGE FINAL CONVERGED VALUE	79.45	80.63	32.14	15.8550
VARIANCE WHEN CONVERGE BELOW TO 0.001 OR 0.1	2.3912×10^{-6}	2.6771×10^{-6}	5.7116×10^{-2}	/
VARIANCE AFTER 1000(200) ITERATION TIMES	8.3805×10^{-65}	1.5491×10^{-63}	9.8326×10^{-34}	1.4409×10^{-11}

Table(2-4-2-1) the running condition of **CK** after combined with SAA



Figure(2-4-2-1-1) the running result of the average iteration times of **CK** with SAA for the test function



Figure(2-4-2-1-2) the running result of the average iteration times of **CK** with and without SAA for the test function

According to the Table(2-4-2-1) and Figure(2-4-2-1-1), Figure(2-4-2-1-2), there is significant improvement in multimodal functions, but shown in Figure(2-4-2-1-2), the convergence speed of *Griewank* becomes even slower. This indicates SAA does not converge as fast as before in the early stage but ensures the success rate or convergence accuracy of the multimodal functions in some ways. Through the close to 1, which says its success rate is guaranteed.

2-4-3 Cauchy Mutation

To study the performance of the Cauchy mutation, we start with confirming the value of variable η in four test functions by comparing the average number of iterations, variance when converge to the success standard, final variance and convergence speed with each other. Because of too much data and figures, for detailed experimental data, please refer to *Appendix* **1**.

From the diagrams, we can draw the following conclusions:

1. For the unimodal functions, the average iteration times decrease and average variances increase with the increase of η generally.
2. For the final variances of the unimodal functions, their levels keep low when η

equals to 1 to 6 approximately and they surge when η is greater than 7.

3. For *Griewank*, the change of η has little effect on the increase or decrease of the average iteration times and the average variances and their levels keep low.
4. For *Rastrigrin*, the minimum value that the particles searched and average variances after 200 iteration times increase with the increase of η in general, but their curves fluctuate more significant than the unimodal ones.
5. The change of η has little effect on the convergence speed whether in unimodal functions and multimodal functions.
6. Specially, the adaptability of *Griewank* will slump at about 15 to 35 iteration times.
7. For the iteration times of the unimodal functions, comparing with the algorithm in 2-4-2, they can converge to certain precision(0.001) with fewer iteration times when η is greater than about 10.
8. For the unimodal functions, the staged variances (when converge below 0.001) are much higher than the algorithm in 2-2. For *Sphere*, the final variances is lower than 2-4-2 when η is at about 1 to 5. However, as for *Step* the final variances are much higher than 2-2 regardless the value of η.
9. For *Griewank*, the iteration times and the variances are basically equal to the algorithm in 2-4-2 regardless the value of η.
10. For *Rastrigrin*, the iteration times are basically equal to the 2-2 when η equals about 1 to 11, and it keeps going higher when η is bigger. And its variances is much bigger than 2-4-2 regardless the value of η.

In conclusion, for different values of η, Cauchy mutation gives different degrees of optimization or regression for unimodal and multimodal functions, comparing with one in 2-4-2 and standard PSO. The selection of the value of η ought to be determined according to the specific situation, but in general, Cauchy mutation has an optimization effect on the entire algorithm.

III. Theoretical Analysis of the Algorithm

Compared with other swarm intelligence algorithms, PSO has more concise features. Meanwhile, the performance of PSO also depends on the parameter setting.[19] Better parameter setting can greatly improve the convergence success rate of PSO, so as to avoid falling into local optima and stagnation as much as possible.

In order to prove the importance of the parameters for PSO, in the **first** part of this section the influence of three parameters on the performance of the algorithm will be introduced. In the **second** part ANOVA will be conducted to judge whether the changes of each parameters has a significant impact on the performance of **DSCPSO**. In the **last** part of this section, convergence analysis of **DSCPSO** proposed in Section II will be performed to show that the proposed algorithm is globally convergent.

3-1 Parametric Influence

The parameters of PSO mainly include speed range, individual learning factor($c1$), group learning factor($c2$), inertia factor(ω) or contraction factor(φ), etc. The settings of different parameters have different effects and influences on the algorithm.

(1)The settings of speed range. Eq(1-1') contains random variables, that cause position update Eq(1-2) numerically uncontrollable. In order to limit this irregular beating, we introduce a speed range $[-v_{max}, v_{max}]$ to limit the speed of its beating. If v_{max} is larger, it is more conducive to the global search, but it is easy to jump over the optimal solution and become stagnant. Similarly, if v_{max} is smaller, it is more conducive to local search, but because the jump is too small, it is easy to fall into local optima.[20][21] The value of v_{max} is mainly determined by empirically adjusting the parameters, and some paper points out that v is generally set to 10%~20% of the problem space.[20]

(2)The settings of learning factors. The learning factors $c1$ and $c2$ are used to control the movement of the particles to the personal best position $(pbest)$ and game best position $(gbest)$. Similar with the settings of speed range, If $c1$ is large,

it is more conducive to the global search, but it will slow down the convergence speed. If $c2$ is large, it is more conducive to accelerate the convergence speed, but it will be easier to lead the particles fall into local optima. For their respective disadvantages, Researchers have proposed linear decreasing or increasing strategies (similar with **Lin** mentioned in Section II) and strategies to increase population diversity, which have solved the above problems to the greatest extent.

(3) inertia factor and contraction factor. As introduced in Section I, researchers proposed inertia factor to optimize PSO. Some paper suggests that the inertia factor should decrease linearly from 0.9 to 0.4.[22] Dos Santos et al. adopted the stochastic approximation theory and proposed a strategy in which the inertia factor decreases to 0 with the time of iterations. PSO with contraction factor was proposed by Kennedy and Clerc[14], its basic form is

$$v_{ij}t = xv_{ij}t - 1 + c1r1(p_{ij} - x_{ij}t) + c2r2(t)(p_{ij} - x_{ij}t) \qquad (3\text{-}1\text{-}1)$$

where $x = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$, $\varphi = c1 + c2$.

The method **CK** mentioned in Section II is a simplified version of it, avoiding imaginary number due to small $c1$ and $c2$ and directly regard x as the ω according to the Eq(1-1'). According to the experiment in the Section II, it is proved that its simplified version is also effective for algorithm optimization.

Although the PSO with inertia factor and contraction factor has its own advantages, because the PSO with inertia factor often adopts a decreasing strategy, the inertia weight is too small in the later stage, and part of the global search ability is lost, while the PSO with contraction factor there is no such deficiency.[23]

3-2 ANOVA

ANOVA is used to deal with the comparison of multiple population means and is one of the important methods in statistical analysis. This method can be used to analyze the differences in the influence of different levels of the same parameter or

the various levels of different parameters on the performance of the algorithm, so as to explore the potential relationship between different parameter setting ranges and the algorithm system.

One-way ANOVA is to analyze the influence of a factor on the whole experiment by observing the change of a factor. In this paper, the same number of experiments was used to conduct ANOVA. In order to analyze the influence of each of ω, $c1$ and $c2$ on the algorithm, 3 separate ANOVAs will be performed here and conduct 6 levels per group (0.3,0.6,0.9,1.2,1.5,1.8. When one parameter is analyzed by ANOVA, the other two parameters are taken as 0.9) and do 5 experiments per level.

For convenience of marking, suppose the factor of ω, $c1$ and $c2$ are A, B, C respectively. Take factor A as the example (B and C are same with A). Let A has m levels, do k experiments per level, then a fitness value can be obtained after each experiment, which denoted as x_{ij}, where $i = 1,2,3,\dots,m$ and $j = 1,2,3,\dots,k$. When analyzing the influence of the change of factor A on the algorithm, the m levels of A $(A_1, A_2, \dots, A_m)$ are regarded as m normal populations. Therefore it can be assumed $X_{ij} \sim N(\mu_i, \sigma^2)$, $i = 1,2,3,\dots,m$, $j = 1,2,3,\dots,k$ and $\mu_i = \mu + a_i$, where μ is the total mean and a_i is the main effect due to the ith level A_i of factor A, which means if to test whether there is a significant difference between the levels of A, test the following hypothesis:

$$H_0: \mu_1 = \mu_2 = \cdots = \mu_m$$

Its alternative hypothesis is:

$$H_1: \mu_1, \mu_2, \dots and\ \mu_m\ are\ not\ all\ equal$$

If H_0 holds, the mean of each level of factor A is the same, and there is no significant difference among m levels of factor A. On the contrary, it is said that there is a significant difference between the m levels of factor A. As can be seen from the above, there $m = 6$, $k = 5$.

(1) ANOVA for Factor A

For function *Sphere* and *Griewank*, let $c1 = c2 = 1.5$; $\omega_1 = 0.3, \omega_2 = 0.6, \omega_3 = 0.9, \omega_4 = 1.2, \omega_5 = 1.5, \omega_6 = 1.8$. The number of particles $N = 100$, iteration times $ger = 100$. Conduct one-way ANOVA, run ***program****, each experiment was required to repeat 5 times. The running results are shown in Table(3-5-1-1):

	0.3	**0.6**	**0.9**	**1.2**	**1.5**	**1.8**
1	0.022164	9.3625×10^{-9}	0.0012587	0.63355	1.4721	1.7076
2	0.029767	1.5587×10^{-6}	0.0069404	0.6966	1.4317	2.4417
3	0.011473	6.0527×10^{-10}	0.0027286	0.58875	1.2007	1.1771
4	0.0099417	2.0883×10^{-8}	0.0077735	0.30013	1.4022	1.1132
5	0.015784	7.6303×10^{-8}	0.0038226	0.86972	1.1848	1.9424
SAMPLE SUM	0.01560954	1.6659×10^{-6}	0.0225238	3.08875	6.6915	8.3820
SAMPLE MEAN	0.03121908	3.3317×10^{-7}	0.00450476	0.61775	1.3383	1.6764

Table(3-2-1-1) the running result of *program** for *Sphere*

According to table(3-5-1-1), it is able to calculate its total deviation sum of squares (S_T), error deviation sum of squares (S_e), factor deviation sum of squares $\left(S_{A(B)(C)}\right)$ and their respective degree of freedom(df).

$$S_T = \sum_{i=1}^{m}\sum_{j=1}^{k}\left(x_{ij} - \overline{x_{\cdot}}\right)^2 = 14.907 \quad f_T = mk - 1 = 29$$

$$S_e = \sum_{i=1}^{m}\sum_{j=1}^{k}\left(x_{ij} - \overline{x_{i\cdot}}\right)^2 = 1.507 \quad f_e = mk - m = 24$$

$$S_A = k\sum_{i=1}^{m}(\overline{x_{i\cdot}} - \bar{x})^2 = 13.4 \quad f_A = m - 1 = 5$$

where total deviation sum of squares (S_T) is a data indicator that describes the degree of dispersion of the entire experiment results; error deviation sum of squares (S_e) reflects random fluctuations in samples within a group and factor deviation sum of squares $\left(S_{A(B)(C)}\right)$ reflects data differences caused by different effects (factors). Specially, There is a relationship between the 3 kinds of deviation sum of squares:

$$S_T = S_A + S_e \tag{3-1}$$

Proof:

Note that

$$\sum_{i=1}^{m}\sum_{j=1}^{k}\left(x_{ij} - \overline{x_{i\cdot}}\right)(\overline{x_{i\cdot}} - \bar{x}) = \sum_{i=1}^{m}[((\overline{x_{i\cdot}} - \bar{x}))\sum_{j=1}^{k}\left(x_{ij} - \overline{x_{i\cdot}}\right)] = 0$$

Therefore

$$S_T = \sum_{i=1}^{m}\sum_{j=1}^{k}\left(x_{ij} - \overline{x_{\cdot}}\right)^2 = \sum_{i=1}^{m}\sum_{j=1}^{k}[\left(x_{ij} - \overline{x_{i\cdot}}\right) + (\overline{x_{i\cdot}} - \bar{x})]^2$$

$$= S_e + S_A + 2\sum_{i=1}^{m}\sum_{j=1}^{k}\left(x_{ij} - \overline{x_{i\cdot}}\right)(\overline{x_{i\cdot}} - \bar{x}) = S_e + S_A$$

Similarly, their degrees of freedom have familiar relationship, the proof is obvious.

The above formula shows that the sum of squares of total deviations is composed of the sum of error deviations and the sum of squares of factor deviations. Therefore, formula(3-1) reflects the proportion of the two differences:

$$F = \frac{MS_A}{MS_e} \tag{3-2-1}$$

where $MS = \frac{Q}{f_Q}$, is the mean square in statistics, it means how many sums of squares are on average in each degree of freedom. Thus, formula(3-1)can be rewritten as formula(3-2) in this example:

$$F = \frac{\frac{S_A}{f_A}}{\frac{S_e}{f_e}} = \frac{\frac{S_A}{m-1}}{\frac{S_e}{m(k-1)}} \tag{3-2-2}$$

The larger the F, the more significant the difference caused by each level. Because

$$x_{ij} = \mu_i + \varepsilon_{ij} \tag{3-2-3}$$

thus the statistic F defined by the formula(3-2) follows the F distribution of f_A and f_e. Consider that the larger the value of F, the more it tends to reject the null hypothesis, therefore, the rejection domain of the test is: [24]

$$W = \{F \geq F_{1-\alpha}(f_A, f_e)\} \tag{3-2-4}$$

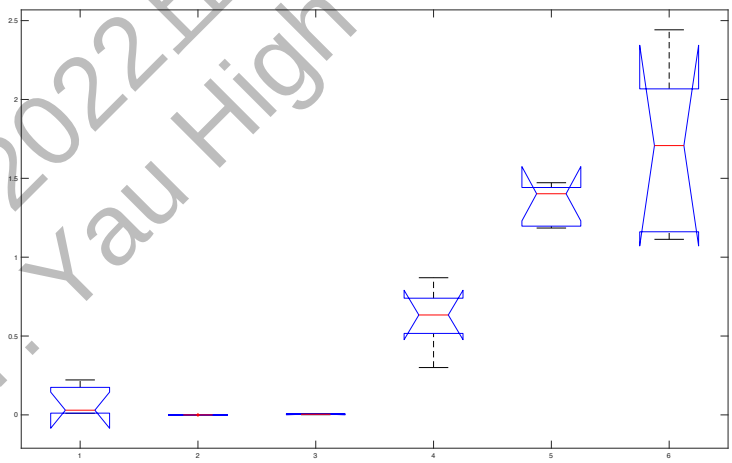
For certain α, we can make the following judgement:

- If $F \geq F_{1-\alpha}(f_A, f_e)$, consider factor A to be significant.
- If $F < F_{1-\alpha}(f_A, f_e)$, consider factor A to be not significant.

Through the above calculation and analysis, we can now list the analysis of variance in a table. The table of factor A is shown in Table(3-5-1-2).

SOURCE	SS	DF	MS	F	PROB>F
FACTOR	13.4	5	2.68	42.68	3.56791×10^{-11}
ERROR	1.507	24	0.06279		
TOTAL	14.907	29			

Table(3-2-1-2) the analysis of variance of factor A for *Sphere*



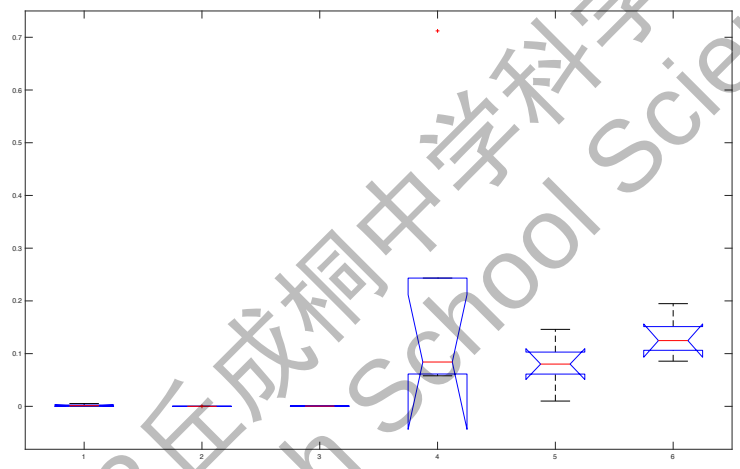
Figure(3-2-1-3) the boxplots for Factor A Analysis for *Sphere*

Set $\alpha = 0.1$, $F_{0.9}(5,24) = 2.10$. Because $F = 42.68 > 2.10$, therefore there is a significant different in the effect of factor $A\ (\omega)$ on PSO performance. $p = 3.56791 \times 10^{-11} < \alpha$, so reject the null hypothesis.

Similarly, for ANOVAs of function *Griewank*, factor B and factor C are all familiar with the one of factor A for *Sphere*. Due to the big number of running data, the running data is shown in *Appendix* **2-1-2**.

SOURCE	SS	DF	MS	F	PROB>F
FACTOR	0.17656	5	0.03531	2.47	0.0612
ERROR	0.34341	24	0.01431		
TOTAL	0.51997	29			

Table(3-2-1-2') the analysis of variance of factor A for *Griewank*



Figure(3-2-1-3') the boxplots for Factor A Analysis for *Griewank*

Set $\alpha = 0.1$, $F_{0.9}(5,24) = 2.10$. Because $F = 2.47 > 2.10$, therefore there is a significant different in the effect of factor $A\ (\omega)$ on PSO performance. $p = 0.0612 < \alpha$, so reject the null hypothesis.

(2)ANOVA for Factor B

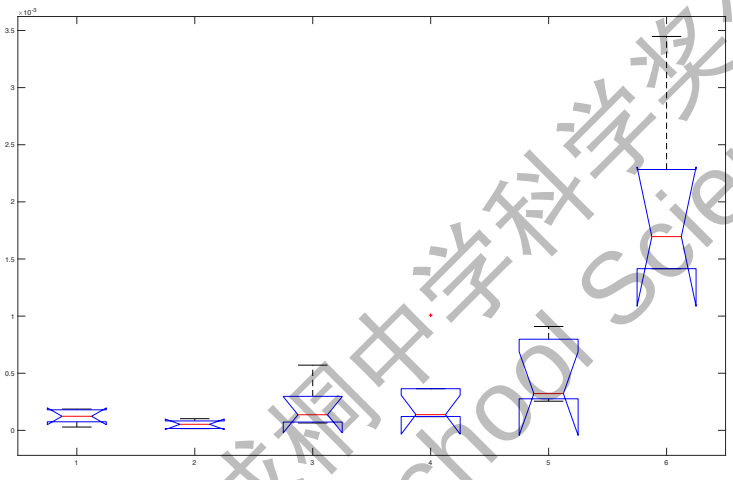
Sphere:

Set $c2 = 0.9, \omega = 0.9; c11 = 0.3, c12 = 0.6, c13 = 0.9, c14 = 1.2, c15 = 1.5, c16 = 1.8$. The running data for *Sphere* is shown in *Appendix* **2-2-1**.

SOURCE	SS	DF	MS	F	PROB>F

FACTOR	1.31548×10^{-5}	5	2.63095×10^{-6}	15.53	7.5072×10^{-7}
ERROR	4.0646×10^{-6}	24	1.69358×10^{-7}		
TOTAL	1.72194×10^{-5}	29			

Table(3-2-2-2) the analysis of variance of factor B for *Sphere*



Figure(3-2-2-3) the boxplots for Factor B Analysis for *Sphere*

Set $\alpha = 0.1$, $F_{0.9}(5,24) = 2.10$. Because $F = 15.53 > 2.10$, therefore there is a significant different in the effect of factor B $(c1)$ on PSO performance. $p = 7.5072 \times 10^{-7} < \alpha$, so reject the null hypothesis.

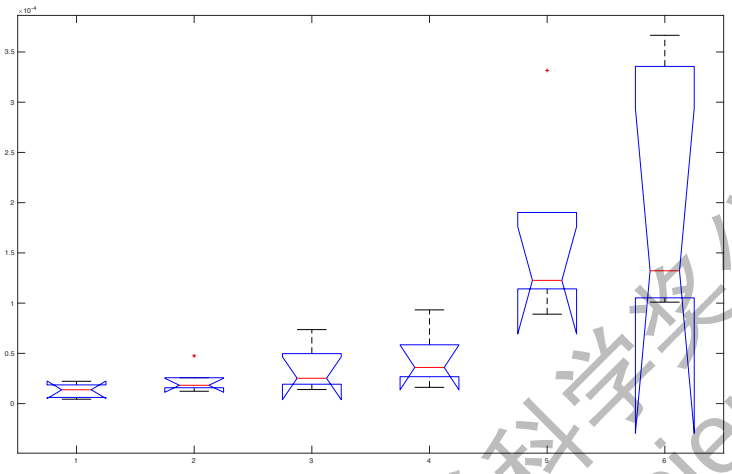
Griewank:

The running data for *Griewank* is shown in *Appendix* **2-2-2**.

SOURCE	SS	DF	MS	F	PROB>F
FACTOR	1.68535×10^{-7}	5	3.3707×10^{-8}	7.31	3×10^{-4}
ERROR	1.10661×10^{-7}	24	4.61089×10^{-9}		

TOTAL	2.79197 $\times 10^{-7}$	29			

Table(3-2-2-2') the analysis of variance of factor B for *Griewank*



Figure(3-2-2-3') the boxplots for Factor B Analysis for *Griewank*

Set $\alpha = 0.1$, $F_{0.9}(5,24) = 2.10$. Because $F = 7.31 > 2.10$, therefore there is a significant different in the effect of factor $B(c1)$ on PSO performance. $p = 3 \times 10^{-4} < \alpha$, so reject the null hypothesis.

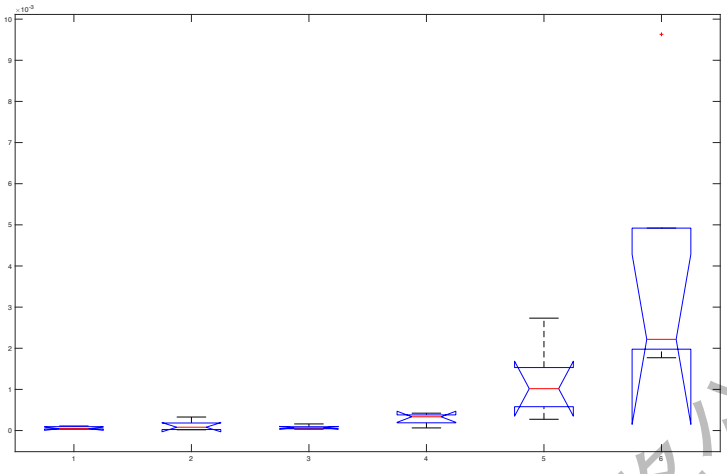
(3)ANOVA for Factor C

Sphere:

Set $c1 = 0.9, \omega = 0.9; c21 = 0.3, c22 = 0.6, c23 = 0.9, c24 = 1.2, c25 = 1.5, c26 = 1.8$. The running data for *Sphere* is shown in *Appendix* **2-3-1**.

SOURCE	SS	DF	MS	F	PROB>F
FACTOR	5×10^{-5}	5	1.08659×10^{-5}	5.48	0.0017
ERROR	5×10^{-5}	24	1.9821×10^{-6}		
TOTAL	1×10^{-4}	29			

Table(3-3-3-2) the analysis of variance of factor C for *Sphere*



Figure(3-3-3-3) the boxplots for Factor C Analysis for *Sphere*

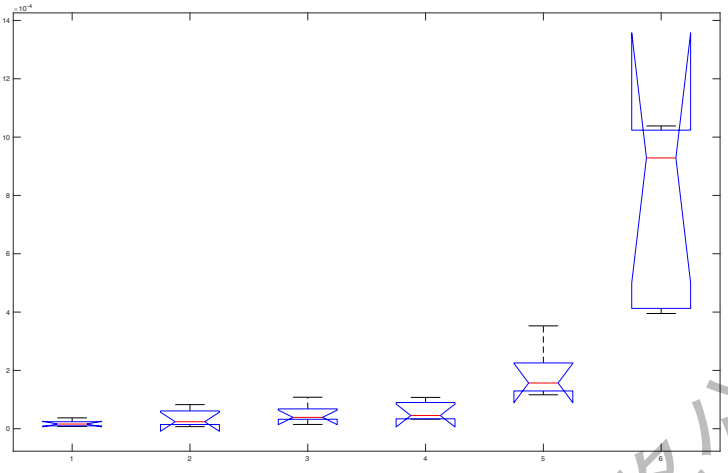
Set $\alpha = 0.1$, $F_{0.9}(5,24) = 2.10$. Because $F = 5.48 > 2.10$, therefore there is a significant different in the effect of factor $C(c2)$ on PSO performance. $p = 0.0017 < \alpha$, so reject the null hypothesis.

Griewank:

The running data for *Griewank* is shown in *Appendix* **2-3-2**.

SOURCE	SS	DF	MS	F	PROB>F
FACTOR	2.06848×10^{-6}	5	4.13695×10^{-7}	21.01	4.67909×10^{-8}
ERROR	4.72556×10^{-7}	24	1.96898×10^{-8}		
TOTAL	2.54103×10^{-6}	29			

Table(3-3-3-2') the analysis of variance of factor C for *Griewank*



Figure(3-3-3-3') the boxplots for Factor C Analysis for *Griewank*

Set $\alpha = 0.1$, $F_{0.9}(5,24) = 2.10$. Because $F = 21.01 > 2.10$, therefore there is a significant different in the effect of factor $C(c2)$ on PSO performance. $p = 4.67909 \times 10^{-8} < \alpha$, so reject the null hypothesis.

The results of the three groups of experiments show that the changes of ω, $c1$ and $c2$ all have significant differences on PSO performance. Review the algorithm content of Section II of this paper, we only uses descent method on ω and $c1$, it can be guessed that the changes of the two also have a significant impact on the performance of PSO. Here may wish to use two-way ANOVA to explore the impact of the two on performance.

The mathematical principles of two-way ANOVA are basically same with the one-way ANOVA's. For two-way ANOVA with interaction, the results of each experiment can be expressed as:

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk} \tag{3-2-5}$$

where μ is the mean value, a_i, b_j and γ_{ij} are the error due to factor A, B and interaction of A and B respectively, ε_{ijk} is the random error at the *i*th and the *j*th level.

Similarly, two-way ANOVA also own its relationship between the 5 kinds of deviation sum of squares:

$$S_T = S_A + S_B + S_{A\times B} + S_e \tag{3-2-6}$$

where

$$S_T = \sum_{q=1}^{k}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(x_{ijq} - \bar{x}\right)^2 \quad f_T = mnk - 1$$

$$S_A = nk\sum_{i=1}^{m}(\overline{x_{i\cdot}} - \bar{x})^2 \quad f_A = m - 1$$

$$S_B = mk\sum_{j=1}^{n}\left(\overline{x_{\cdot j}} - \bar{x}\right)^2 \quad f_B = n - 1$$

$$S_{A\times B} = k\sum_{i=1}^{m}\sum_{j=1}^{n}\left(x_{ij} - \overline{x_{i\cdot}} - \overline{x_{\cdot j}} + \bar{x}\right)^2 \quad f_{A\times B} = (m-1)(n-1)$$

$$S_e = \sum_{q=1}^{k}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(x_{ijq} - \overline{x_{ij}}\right)^2 \quad f_e = mn(k-1)$$

The proof of Eq(3-7) is similar with that of Eq(3-1), so it is omitted here.

(4)ANOVA for Factor A and B

Set $c2 = 0.8$; $\omega_1 = 0.3, \omega_2 = 0.6, \omega_3 = 0.9, \omega_4 = 1.2, \omega_5 = 1.5, \omega_6 = 1.8; c11 = 0.2, c12 = 0.4, c13 = 0.6\ c14 = 0.8, c15 = 1.0, c16 = 1.2.$ Do 3 two-way ANOVA experiments at each level of factor A and B (hence $k = 3, m = n = 6$). Test the following hypothesis:

$$H_{01}: \alpha_1 = \alpha_2 = \cdots = \alpha_i$$

$$H_{02}: \beta_1 = \beta_2 = \cdots = \beta_j$$

$$H_{03}: \gamma_{11} = \gamma_{12} = \cdots = \gamma_{21} = \gamma_{22} = \cdots = \gamma_{ij}$$

Their alternative hypothesis are:

$$H_{11}: \alpha_1, \alpha_2, \ldots, \alpha_n \ are\ not\ all\ equal$$

$$H_{12}: \beta_1, \beta_2, \ldots, \beta_n \ are\ not\ all\ equal$$

$$H_{13}: \gamma_{11}, \gamma_{12}, \ldots, \gamma_{ij} \ are\ not\ all\ equal$$

The running data for *Sphere* is shown in *Appendix* **2-4:**

SOURCE	SS	DF	MS	F	PROB>F
FACTOR A	60.92381	5	12.18476	153.3047	6.2175×10^{-37}
FACTOR B	1.93818	5	0.38764	4.87712	6.7311×10^{-4}
INTERACTION EFFECT	5.39999	25	0.21600	2.71764	5.1304×10^{-4}
ERROR	5.72261	72	0.07948		
TOTAL	73.98459	107			

Table(3-2-4-2) the analysis of variance of factor A and B for *Sphere*

Set $\alpha = 0.1$, $F_{0.9}(5,72) = 1.93$. Because $F_A = 153.3047 > 1.93$, $F_B = 4.87712 > 1.93$, $F_{A\times B} = 2.71764 > 1.93$, therefore there is a significant different in the effect of factor $A(\omega)$, $B(c1)$ and $'A \times B'$ on PSO performance. $p_A = 6.2175 \times 10^{-37} < \alpha$, $p_B = 6.7311 \times 10^{-4} < \alpha$, $p_{A\times B} = 5.1304 \times 10^{-4} < \alpha$, so reject the null hypothesis.

The effect of the two-factor ANOVA on the performance of PSO under the *Griewank* function is omitted here. It can be expected that all three factors' effect have a significant different on PSO performance.

Additionally, when the test results are significant, we can further find out total mean μ, each horizontal effect a_i and error variance σ^2.

Because $X_{ij} \sim N(\mu, \sigma^2)$, therefore the likelihood function is:

$$L(\mu, a_1, a_2, a_3, a_4, a_5, a_6, \sigma^2) = \prod_{i=1}^{6}\prod_{j=1}^{5}\left\{\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(y_{ij}-\mu-a_i)^2}{2\sigma^2}\right]\right\} \quad (3\text{-}2\text{-}7)$$

Its Log-Likelihood function is:

$$l(\mu, a_1, a_2, a_3, a_4, a_5, a_6, \sigma^2) = -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{6}\sum_{j=1}^{5}\left(y_{ij} - \mu - a_i\right)^2 \quad (3\text{-}2\text{-}8)$$

where $n = mk = 30$.

Then the likelihood equations (Eq3-2-9) are:

$$\begin{cases} \dfrac{\partial l}{\partial \mu} = \dfrac{1}{\sigma^2}\displaystyle\sum_{i=1}^{6}\sum_{j=1}^{5}\left(y_{ij} - \mu - a_i\right)^2 = 0 \\ \dfrac{\partial l}{\partial a_i} = \dfrac{1}{\sigma^2}\displaystyle\sum_{j=1}^{5}\left(y_{ij} - \mu - a_i\right) = 0 \\ \dfrac{\partial l}{\partial \sigma^2} = -\dfrac{n}{2\sigma^2} + \dfrac{1}{2\sigma^4}\displaystyle\sum_{i=1}^{6}\sum_{j=1}^{5}\left(y_{ij} - \mu - a_i\right)^2 = 0 \\ \displaystyle\sum_{i=1}^{6} a_i = 0 \end{cases}$$

Solve the equations(3-2-9), the maximum likelihood estimation of each parameter can be found out:

$$\begin{cases} \hat{\mu} = \bar{y} \\ \widehat{a_i} = \overline{y_{i\cdot}} - \bar{y}, \ i = 1,2,,\ldots,6 \\ \widehat{\sigma_M}^2 = \dfrac{1}{n}\displaystyle\sum_{i=1}^{6}\sum_{j=1}^{5}\left(y_{ij} - \overline{y_{i\cdot}}\right)^2 = \dfrac{S_e}{n} \end{cases}$$

Specially, because $\widehat{\sigma_M}^2$ is not the unbiased estimation of σ^2, we usually take Eq(3-2-10) into practical problem solving:[24]

$$\hat{\sigma}^2 = S_e = \frac{S_e}{f_e} \quad (3\text{-}2\text{-}10)$$

According to the running data shown in *Appendix* **2-1-1** to **2-3-2**, let their subscript codes be A, A', B, B', C and C' respectively, the value of $\hat{\mu}$, $\hat{a}$ and $\hat{\sigma}^2$ of each circumstance can be found. Only part of the calculation results are shown here. The detailed value can be seen in *Appendix* **3-1**.

$$\widehat{\mu_A} = 0.611362362195, \ \widehat{\mu_{A'}} = 0.05176453825,\ldots$$

$$\widehat{a_{A1}} = -0.580143282195, \ \widehat{a_{B1}} = -3.6163 \times 10^{-4}, \ \widehat{a_{C1}} = -8.5111 \times 10^{-4},\ldots$$

$$\widehat{\sigma_A}^2 = 0.062792,\ldots$$

Next, confidence intervals for each level of the factor A, B and C are discussed below.

Because $\overline{y_{i\cdot}} \sim N(\mu_i, \frac{\sigma^2}{k})$, $\frac{S_e}{\sigma^2} \sim \chi^2(f_e)$ and both of them are independent, therefore

$$\frac{\sqrt{k}(\overline{y_{i\cdot}} - \mu_i)}{\sqrt{\frac{S_e}{f_e}}} \sim t(f_e) \qquad (3-2-11)$$

According to formula(3-2-11), the confidence interval for the level mean of A_i for $1-\alpha$ for μ_i is:

$$\left[\overline{y_{i\cdot}} \pm \frac{t_{1-\frac{\alpha}{2}}(f_e)\hat{\sigma}}{\sqrt{k}}\right] \qquad (3-2-12)$$

where $\hat{\sigma}$ is given in Eq(3-11). Before we have obtained $\widehat{\mu_A} = 0.611362362195$, $\widehat{\mu_{A'}} = 0.05176453825$, $\widehat{\mu_B} = 5.3532 \times 10^{-4}$, $\widehat{\mu_{B'}} = 8.0576 \times 10^{-5}$, $\widehat{\mu_C} = 0.00913456$, $\widehat{\mu_{C'}} = 1.8589 \times 10^{-4}$; $\widehat{\sigma_A}^2 = 0.062792$, $\widehat{\sigma_{A'}}^2 = 0.014309$, $\widehat{\sigma_B}^2 = 1.69535 \times 10^{-5}$, $\widehat{\sigma_{B'}}^2 = 4.6109 \times 10^{-7}$, $\widehat{\sigma_C}^2 = 2.0833 \times 10^{-4}$, $\widehat{\sigma_{C'}}^2 = 1.9689 \times 10^{-6}$ (shown in *Appendix* **3-1**), set $\alpha = 0.1$, $t_{1-\frac{\alpha}{2}}(f_e) = t_{0.95}(24) = 1.7109$, then the value of μ_A, $\mu_{A'}$; μ_B, $\mu_{B'}$; μ_C and $\mu_{C'}$ can be figured out. Also, only part of the calculation results are shown here. The detailed values are shown in *Appendix* **3-2**.

$$\mu_A: \left[0.6113623 \pm 1.7109 \times \frac{\sqrt{0.062792}}{\sqrt{5}}\right] = [0.4992982, 0.7234266]$$

$$\mu_{A'}: \left[0.0517645 \pm 1.7109 \times \frac{\sqrt{0.014309}}{\sqrt{5}}\right] = [0, 0.14903245]$$

…

3-3 Convergence Analysis

This part mainly discuss this problem:[25]

$$\min f(x) \quad (x \in S \in R^n) \qquad (3-3-1)$$

However, in this situation, if the function is discontinuous and the discontinuity measure is 0, for example:

$$f = \begin{cases} x^2 \ (x \neq 1) \\ -1 \ \ (x = 1) \end{cases} \tag{3-3-2}$$

Obviously, the minimum value of f is -1 when $x = 1$, but its measure $\mu(x = -1) = 0$, therefore it is impossible for whether PSO or DSCPSO to find the best solution. In order to avoid such circumstance, we rewrite problem(3-3-1) in (3-3-3):

$$\psi = \inf\{x | v(z \in S | f(z) < x) > 0\} \tag{3-3-3}$$

Because the Lebesgue measure $v(x|f(z) < x) > 0$, it can avoid the circumstance above.

In order to express the detail convergence criterion of random algorithms, their basic structure is shown below

$$Step\ 0: Randomly\ choose\ the\ start\ point\ z_0 \in S, set\ k = 0;$$

$$Step\ 1: Generate\ vector\ \overrightarrow{\xi_k}\ from\ Sample\ Space(R^n, B, \mu_k)$$

$$Step\ 2: Calculate\ z_{k+1} = I\left(z_k, \overrightarrow{\xi_k}\right), choose\ \mu_{k+1}, let\ k = k + 1\ and\ return\ to\ Step\ 1$$

where B is a $\sigma - field$ of a subset of R^n, μ_k is the probability measure in B and (R^n, B, μ_k) represents the probability space of the algorithm at the kth iteration time. Function I is the way that the algorithm iterates.

Definition 3-3-4:

Set M_k is a subset of R^n, then call it the support set of μ_k if it satisfies the restrictions below:

$$\mu_k(M_k) = 1$$

$$Take\ any\ point\ column\ \{y_k\}_{k=1}^{+\infty} \subseteq M_k, for\ any\ of\ its\ convergent\ subcolumns \{y_j^k\}_{k=1}^{+\infty}, satisfy \lim_{j \to +\infty} y_j^k \in M_k$$

$$If\ \forall N \subseteq R^n\ and\ satisfies\ the\ 2\ restrictions\ above, then\ M_k \subseteq N$$

Additionally, a random algorithm ought to satisfy the hypothesis below:

Hypothesis 1:

$$f\left(I\left(z,\vec{\xi}\right)\right) \leq f(z)$$

and if $\vec{\xi} \in S$, then

$$f\left(I\left(z,\vec{\xi}\right)\right) \leq f(\vec{\xi})$$

Hypothesis 2:

For any Borel Subset A of S, if its measure $v(A) > 0$, then

$$\prod_{k=0}^{\infty}\left(1-\mu_k(A)\right) = 0$$

where $\mu_k(A)$ is the probability that get A from measure μ_k.

Moreover, a sufficient and necessary condition for the global convergence of the random algorithm can be given by Hypothesis 1 and 2:

Theorem 3-3-5:

Suppose f is a measurable function, area S is a measurable subset and Hypothesis 1, 2 satisfy. Set the algorithm-generated solution sequence is $\{z_k\}_{k=1}^{+\infty}$, which satisfy

$$\lim_{k\to+\infty} P(z_k \in R_\varepsilon) = 1$$

where R_ε is the $\varepsilon - acceptable\ field$ of the algorithm, which is expressed as

$$R_\varepsilon = \{z \in S | f(z) < \psi + \varepsilon\}$$

where $\varepsilon > 0$; and $P(z_k \in R_\varepsilon)$ is the solution generated by the algorithm at the kth iteration time.

Proof:

From Hypothesis 1, if $z_k \in R$ or $\overrightarrow{\xi_k} \in R$, then for $\forall k' > k$, satisfy $z_{k'} \in R_\varepsilon$, therefore

$$P(z_k \in R_\varepsilon) = 1 - P(z_k \in S \backslash R_\varepsilon) \geq 1 - \prod_{i=0}^{k-1}\left(1-\mu_i(R_\varepsilon)\right) \quad (3-3-6)$$

Take the limit of $P(z_k \in R_\varepsilon)$ and $\prod_{i=0}^{k-1}\left(1-\mu_i(R_\varepsilon)\right)$, then

$$1-\lim_{k\to+\infty}\prod_{i=0}^{k-1}\left(1-\mu_i(R_\varepsilon)\right) \le \lim_{k\to+\infty} P(z_k \in R_\varepsilon) \le 1 \qquad (3-3-7)$$

From Hypothesis 2, because

$$\prod_{k=0}^{\infty}\left(1-\mu_k(A)\right)=0$$

therefore

$$\lim_{k\to+\infty} P(z_k \in R_\varepsilon)=1 \qquad (3-3-8)$$

3-3-1 Convergence Analysis of the Standard PSO

Theorem 3-3-1-1:

The standard PSO satisfies Hypothesis 1.

Proof:

For the function I:

$$I(p_{g,k}, x_{i,k}) = \begin{cases} p_{g,k} & f(p_{g,k}) \le f(x_{i,k}) \\ x_{i,k+1} & f(p_{g,k}) > f(x_{i,k}) \end{cases}$$

where $x_{i,k+1} = x_{i,k} + \omega v_{i,k} + c_1 rand_1\left(pbest_i - x_{i,k}\right) + c_2\left(gbest - x_{i,k}\right)$ and $x_{i,k}$ represents the location of the *i*th particle at the *k*th iteration time. Obviously, the standard PSO satisfy Hypothesis 1.

Theorem 3-3-1-2:

The standard PSO do not satisfy Hypothesis 2.

Proof:

The standard PSO satisfies formula (3-3-9) if it satisfies Hypothesis 2:

$$S \subseteq \bigcup_{i=1}^{N} M_{i,k} \qquad (3-3-9)$$

where $M_{i,k}$ represents the support set of the *i*th particle at the *k*th iteration time.

Because

$$V(t) = X(t) - X(t-1)$$

$$X(t+1) = X(t) + wV(t) - X(t)(\phi_1 + \phi_2) + P\phi_1 + P_g\phi_2$$

Therefore

$$M_{i,k} = x_{i,j,k-1} + w\left(x_{i,j,k-1} - x_{i,j,k-2}\right) + \phi_1\left(P - x_{i,j,k-1}\right) + \phi_2(P_g - x_{i,j,k-1})$$

(3-3-10)

where $x_{i,j,k-1}$ represents the value of the *j*th dimension of the *i*th particle at the *k*th iteration time; $0 \le \phi_1 \le c_1$, $0 \le \phi_2 \le c_2$. It is obvious that $M_{i,k}$ represents a hyper-rectangle confirmed by ϕ_1 and ϕ_2.

When $\max\left(c_1\left|P - x_{i,j,k-1}\right|, c_2\left|P_g - x_{i,j,k-1}\right|\right) < \frac{1}{2}diam_j(S)$ holds, it is clear that $v\left(M_{i,k} \cap S\right) < v(S)$, where $diam_j(S)$ represents the length of S at the *j*th dimension. Because $x_i \to \frac{c_1P + c_2P_g}{c_1 + c_2}$[25], therefore $\lim_{k \to +\infty} M_{i,k} = 0$, which says with the increase of iteration times k, $v\left(M_{i,k}\right)$ and $v(\bigcup_{i=1}^{t} M_{i,k})$ are decreasing, which infer to $v\left(\bigcup_{i=1}^{t} M_{i,k} \cap S\right) < v(S)$. This indicates that $\exists k' \in Z$, when $k' > k$, there $\exists subset\ A \in S$, let

$$\sum_{i=1}^{N} \mu_{i,k}(A) = 0 \qquad (3-3-10)$$

which means

$$\prod_{k=0}^{\infty}\left(1 - \mu_k(A)\right) = 1 \neq 0$$

Therefore the standard PSO satisfies Hypothesis 1 but do not satisfy Hypothesis 2, which indicates that the standard PSO is not a global convergence algorithm.

3-3-2 Convergence Analysis of DSCPSO

Theorem 3-3-2-1:

DSCPSO do not satisfy Hypothesis 1.

Proof:

Because DSCPSO combines SAA and Cauchy mutation, on the one hand, according to formula (2-2-1), there is always a probability greater than 0 for the particles to accept a worse solution. On the other hand, according to formula (2-3-3), the particles will randomly oscillate to a position, it will be better or worse. Therefore, for DSCPSO, it cannot ensure

$$f\left(I\left(z,\vec{\xi}\right)\right) \leq f(z)$$

which says DSCPSO do not satisfy Hypothesis 1.

Theorem 3-3-2-2:

DSCPSO satisfies Hypothesis 2.

Proof:

DSCPSO is an algorithm generated by PSO combining with SAA and Cauchy mutation. Due to the randomness of Metropolis criterion and Cauchy mutation, let the union of the support sets for all particles be α, $\exists k' > k$, makes $\alpha \in S$.. Therefore when $v(A) > 0$

$$\sum_{i=1}^{N} \mu_{i,k}(A) = 1$$

which indicates

$$\prod_{k=0}^{\infty}\left(1 - \mu_k(A)\right) = 0$$

Therefore DSCPSO do not satisfy Hypothesis 1 but satisfies Hypothesis 2, which indicates that DSCPSO is not a global convergence algorithm.

Because DSCPSO do not satisfy Hypothesis 1, it is not a local convergence algorithm as well.[25]

IV. The Application - Nucleic Acid Sample Collection Scheduling Optimization Problem

4-1 The Background

At the end of 2019, the global COVID-19 epidemic first broke out in Wuhan and quickly spread to the whole country and the world, bringing a very large negative impact on the economy, politics, people's livelihood and other aspects of countries around the world. In order to curb the further spread of the epidemic, the Chinese government has implemented a number of measures including regional closure management, nationwide material and human support, vaccine and nucleic acid detection kit research and development. Among them, the display index of the nucleic acid kit is an important basis for whether the reference individual is positive or not.

China's nucleic acid kits were developed as early as January 2020 and were used in the most severely affected areas at that time. With the development of productivity and the continuous maturity of related technologies, the scope of using nucleic acid kits developed in China has been continuously expanded and gradually developed to the whole country. Guo Yanhong, the supervisory commissioner of the National Health and Medical Commission's Medical Administration and Hospital Administration Bureau, said at the press conference of the State Council's joint prevention and control mechanism that as of May 13, 2022, there are 13,000 medical and health institutions nationwide that can carry out nucleic acid testing, with 153,000 professional The technicians are engaged in nucleic acid detection technology. The nucleic acid detection capacity has reached 57 million tubes per day. The nucleic acid detection capacity has been significantly improved.[26]

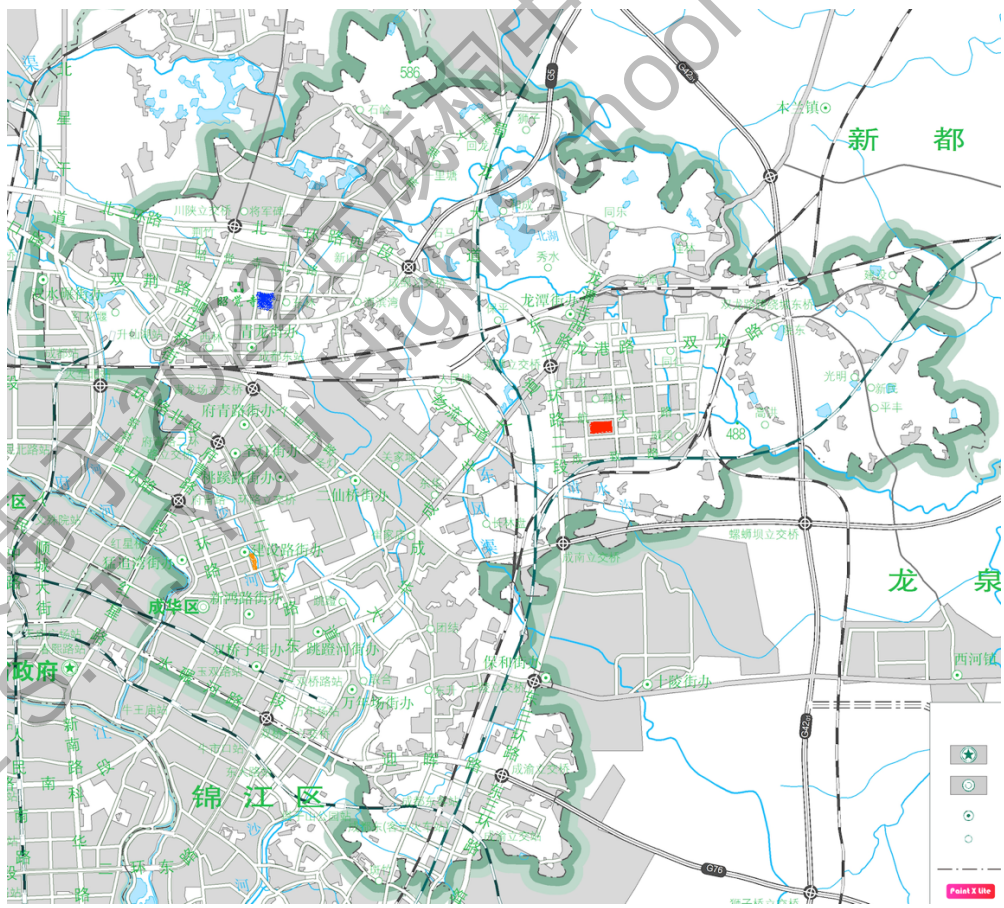
In July this year, the epidemic broke out again in Chenghua District, Chengdu. The district government quickly organized and carried out all-staff accounting and testing, which effectively curbed the spread of the epidemic. In this section of the paper, we will study the optimal scheduling problem of nucleic acid specimen collection, and

analogize it to the Traveling Salesman Problem, and use the improved PSO which has been proposed in the earlier section to solve the problem.

4-2 The Mathematical Model Establishment and Solution

4-2-1 Collection Vehicles Scheduling Selection

This section mainly studies the collection and scheduling of nucleic acid samples in Chenghua District, Chengdu. Therefore, the location information of nucleic acid testing points and testing centers is collected from the official government website. However, due to the large number of nucleic acid detection points, this article will divide the scope of nucleic acid detection points according to the geographical location of each street. The locations of the streets and nucleic acid testing centers in Chenghua District are shown in Figure(4-2-1-1).



Figure(4-2-1) the location of the streets and nucleic acid testing centers in Chenghua District

(The sources are from http://scdfz.sc.gov.cn/scdqs/szdq/cds/chq/content_17758 and

https://mp.pdnews.cn/Pc/ArtInfoApi/article?id=13862572)

where the location of each street (subject to the location of the street office) is marked as ⊙ and there are 3 nucleic acid testing centers, they are Sichuan Jinyu Medical Laboratory Center (painted in red), Chengdu Sixth People's Hospital (painted in orange) and Chengdu Chenghua District Center for Disease Control and Prevention (painted in blue).

According to the information from government website, there are 14 streets in Chenghua District[27]. Because the range of Qinglong and Longtan is much bigger than others, therefore set up two nucleic acid sample collection points in each of these two streets and set one in other street. See Table(4-2-1-2) and Figure(4-2-1-4) for the numbers and coordinates of nucleic acid sample collection points in each sub-district office, and see Matrix(4-2-1-3) for the distances between collection points: (Due to the large number of collection points, only part of the data is displayed. The detail can be seen in *Appendix* **4**)

Number	Location	Coordinate
1	Mengzhuiwan	104.10236E , 30.67693N
2	Shuangqiaozi	104.11259E , 30.65368N
3	Bailianchi	104.14495E , 30.73159N
4	Jianshe Road	104.07275E , 30.57899N
…		
15	Longtan (1)	104.17110E , 30.71094N
16	Longtan (2)	104.17906E , 30.71167N
(17)	Sichuan Jinyu Medical Laboratory Center	104.17518E , 30.69329N
(18)	Chengdu Sixth People's Hospital	104.11468E , 30.67275N

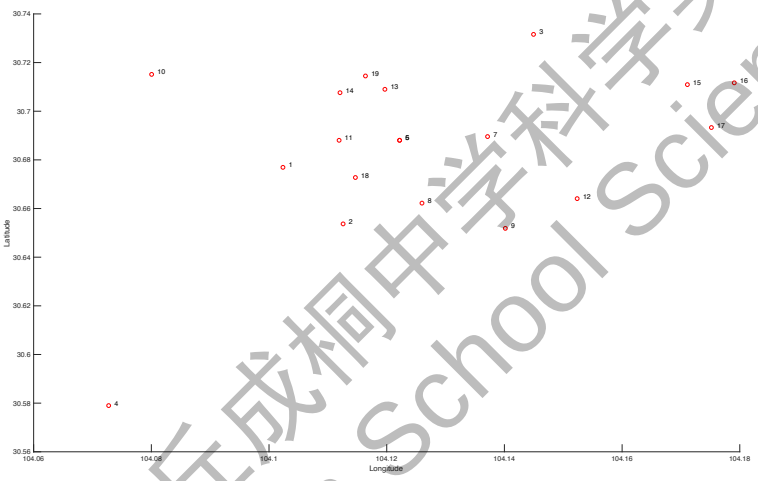
(19)	Chengdu Chenghua District Center for Disease Control and Prevention	104.11639E , 30.71449N

Table(4-2-1-2) the numbers and coordinates of nucleic acid sample collection points in each street

$$\begin{bmatrix} 0 & 3.4 & 11 & 2.3 & \dots & 11 & 1.7 & 6.7 \\ 3.4 & 0 & 13 & 3.1 & \dots & 9.7 & 2.2 & 9.2 \\ 11 & 13 & 0 & 11 & \dots & 8.1 & 11 & 4.3 \\ 2.3 & 3.1 & 11 & 0 & \dots & 8.6 & 1.3 & 6.2 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 11 & 9.7 & 8.1 & 8.6 & \dots & 0 & 9.9 & 9.1 \\ 1.7 & 2.2 & 11 & 6.3 & \dots & 9.9 & 0 & 7.2 \\ 6.7 & 9.2 & 4.3 & 1.2 & \dots & 9.1 & 7.2 & 0 \end{bmatrix}$$

Matrix(4-2-1-3) the distances between collection points (unit: km)

where element x_{ij} presents the distance between point i and point j.



Figure(4-2-1-4) Nucleic acid sample collection point coordinate scatter plot

4-2-2 Model Assumptions

In order to transform the sub-scheduling model into a mathematical model, where model assumptions are made to ensure accuracy and rigor.

1. Three nucleic acid specimen collection vehicles depart from three nucleic acid testing centers (point 17, point 18 and point 19) respectively;
2. Each collection vehicle can freely choose the collection point it will go to. Each collection point must have one and only one collection vehicle passing by;
3. Set the velocity of each collection vehicle is $V = 0.5km/min$ and stop for 3 minutes at each collection point;
4. Each collection vehicle must eventually return to the original starting point.

5. The collection vehicles do not need to go through a testing center that is not its own original starting point;
6. Due to the different detection efficiencies of the three detection centers, the three final times of point 17, point 18 and point 19 need to be multiplied by 0.98, 1.05 and 1.02 times respectively.

4-2-3 Model Building

Treat this scheduling problem as a 0-1 planning problem, build mathematical model of TSP. Let the subscripts of the three collection vehicles which depart from point 17, point 18 and point 19 be a, b, c. Determine if two collection points are connected for a collection vehicle:

$$x_{pij} = \begin{cases} 1 & collection\ vehicle\ p\ goes\ from\ collection\ point\ i\ to\ collection\ point\ j \\ 0 & else \end{cases}$$

where $p = a, b, c$ and $i, j = 1,2,3,4, \dots ,19$.

The resulting planning model is as follows:

$$\text{Min } T = 0.98 \sum_{i \neq j}^{19} d_{ij} x_{aij} + 1.05 \sum_{i \neq j}^{19} d_{ij} x_{bij} + 1.02 \sum_{i \neq j}^{19} d_{ij} x_{cij} \qquad (4-1)$$

$$\sum_{j=1}^{19} x_{pij} = 1 , p = a, b, c, i = 1,2,3, \dots ,16 \qquad (4-2)$$

$$\sum_{i=1}^{19} x_{pij} = 1 , p = a, b, c, j = 1,2,3, \dots ,16 \qquad (4-3)$$

$$\sum_{i,j \in s}^{19} x_{pij} < |s| - 1 , 2 \leq |s| \leq 19 - 1, s \subset \{1,2,3, \dots ,19\}, p = a, b, c \qquad (4-4)$$

where formula(4-1) indicates the goal of finding the minimum time index. Specially, the "time index" here is not length of time in general concept, but refers to the total time or work cost consumed (To express the length of time in common concepts, formula(4-1) can be rewritten to $\text{Min}\{\text{Max}\{0.98 \sum_{i \neq j}^{19} d_{ij} x_{aij} , 1.05 \sum_{i \neq j}^{19} d_{ij} x_{bij} , 1.02 \sum_{i \neq j}^{19} d_{ij} x_{cij}\}\}$). Formula(4-2) and (4-3) show every point must be stopped once and only once. Formula(4-4) shows each point can and can only be used as the starting point and the ending point of the route once.

For the calculation of the running time of each collection vehicle, the following formula is given:

$$t_{p1} = \frac{S_p}{V} \qquad (4-5)$$

$$t_{p2} = \sum_{i \neq j}^{19} 3x_{ij} \qquad (4-6)$$

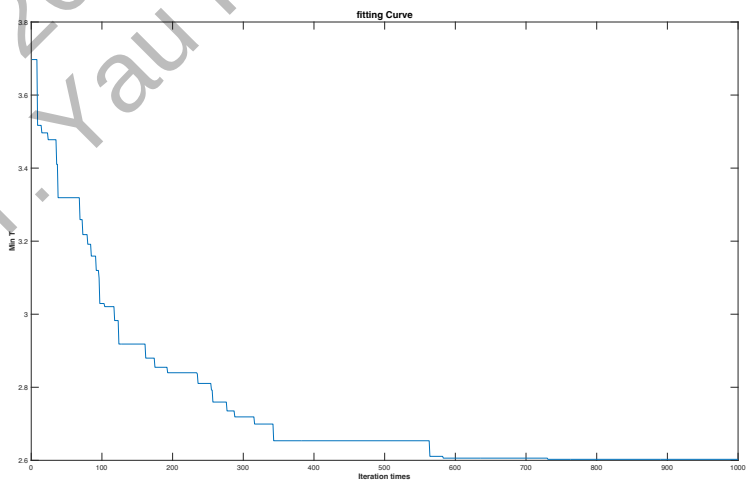
$$T_p = t_{p1} + t_{p2} \qquad (4-7)$$

$$T = T_a + T_b + T_c \qquad (4-8)$$

where formula(4-5) indicates the transport time of the collection vehicle; formula(4-6) indicates the time taken for the collection vehicle to collect specimens at the test point. Time per collection vehicle and total time for three collection vehicles is shown in formula(4-7) and (4-8).

4-2-4 Model Solving Based on the Improved PSO

Set $c10 = 1$, $c2 = 0.1$, $\eta = 1$, iteration times $k = 1000$, $c_{1k} = 0.4$, $T = 2000$, $dT = 0.98$ and particle number $N = 500$. Use **DSCPSO** to solve the problem, and choose **CK** as the descend method. The optimization process and final result are shown in Figure(4-2-4-1) and Table(4-2-4-2).



Figure(4-2-4-1), the curve optimization process

Point number	Vehicle number	Arrival time	Departure time

1	1	49.998	52.998
2	1	32.202	35.202
3	3	8.598	11.598
4	1	57.600	60.600
5	1	66.900	69.900
6	1	63.600	66.600
7	1	73.902	76.902
8	1	39.000	42.000
9	1	23.598	26.598
10	3	37.398	40.398
11	1	80.298	83.298
12	1	15.600	18.600
13	3	20.202	23.202
14	3	25.602	28.602
15	1	91.698	94.698
16	1	97.698	100.698

Table(4-2-4-2) the optimization result

In Table(4-2-4-2), the first column represents the each nucleic acid sample collection point; the second column represents vehicle responsible for sample collection at the point (Collecting vehicles leaving from points 17, 18, 19 are recorded as 1, 2, and 3 respectively); the third column and the fourth column represent the arrival time and departure time of certain collection at the certain point.

According to Figure(4-2-4-1), when the iteration is less than 350 times, the decline speed is fast, and then the speed becomes stable, and the optimal solution is found around the *730th* time. According to Table(4-2-4-2), the optimal scheduling route of this model is:

Vehicle 1: 17→12→9→2→8→1→4→6→5→7→11→15→16→17

Vehicle 2: Idle

Vehicle 3: 19→3→13→14→10→19

Therefore, from the time when the three collection vehicles set off from their respective nucleic acid testing centers to collect nucleic acid samples to the last collection vehicle returning to the starting point, the time index (Min T) is $\mathbf{156.162}$.

4-3 Algorithm Sensitivity Analysis

For the parameter setting of **DSCPSO**, for the number of particles N, in general, the more particles, the stronger the search ability, vice versa; for the temperature drop rate dT, the greater the dT, the greater the probability of accepting a poor solution when performing SAA, vice versa; for the jump distance factor η in Cauchy mutation, the larger the η, the larger the particle jump, which can better prevent falling into the local optimal solution, but it will consume the convergence time, vice versa; for the maximum time of iterations k, if it is too small, it will not be easy to converge to the optimal solution, and if it is too large, it will take more time, so it needs to be set moderately. Study the settings of parameters c_{10}, c_2, c_{1k}, N, η, dT and k, the result is shown in Table(4-3-1).

η	c_{10}	c_{1k}	c_2	N	dT	k	Min T
1	1	0.4	0.1	1000	0.98	1000	**156.162**
2	1	0.4	0.1	1000	0.98	1000	156.948
3	1	0.4	0.1	1000	0.98	1000	156.816
0.5	1	0.4	0.1	1000	0.98	1000	163.956
1	1	0.4	0.1	1000	0.99	500	156.360
1	1	0.4	0.1	1000	0.98	500	158.874
1	1	0.4	1	1000	0.98	1000	181.144
1	0.1	0.04	1	1000	0.98	1000	170.208
1	1	0.4	0.1	1000	0.95	1000	165.288
1	1	0.4	0.1	1000	0.90	1000	165.678
1	2	0.8	0.1	1000	0.98	1000	156.360

1	1	0.4	0.1	500	0.98	1000	**156.162**
1	1	0.4	0.1	500	0.98	1500	156.360
1	1	0.4	0.1	500	0.99	1500	156.360
1	1	0.4	0.1	1500	0.98	1000	**156.162**

Table(4-3-1), the result of different parameter settings

According to Table(4-3-1), for this model, the personal experience of particles needs to be considered more and the group experience needs to be considered less.[28] In order to ensure the iterative effect, it is more reasonable to set the maximum time of iterations to 1000 and to improve the running speed of the program, it is known that the optimal solution can be found by setting the number of particles $N = 500$ and $N = 1000$ when other parameters are the same, therefore $N = 500$ is chosen.

Seen in Table(4-2-4-2), collection vehicle 1 went to the most nucleic acid testing points while collection vehicle 2 was idle. That is because although the nucleic acid testing center 17 is far from the center of the district, it has a lower time multiplier, and although the testing center 18 is closer to the center of the district, because it is a general hospital, nucleic acid specimen testing is not its main character. The time penalty is larger, so less nucleic acid detection points are allocated to it， which shows that the detection efficiency of the nucleic acid testing center has a greater impact on vehicle deployment than its location. Specialized nucleic acid testing institutions should undertake more testing work. However, due to the lack of internal data of nucleic acid testing centers, the testing efficiency of each center is only a guess and this model is only a simplified discussion of nucleic acid sample collection work, so in reality, it should be analyzed in combination with the actual situation, but the above conclusions are still valid.

4-4 Conclusion

This section mainly studies the scheduling optimization of nucleic acid specimen collection vehicles using **DSCPSO**. The path planning model in the traveling

salesman problem, the average speed of the vehicle, the residence time and the distance of each nucleic acid detection point are selected to solve, and the sensitivity of the algorithm is calculated. From the experimental results, through the setting of various parameters, **DSCPSO** successfully solved the optimal scheduling model. The average time of iterations is about 700, and the optimal time index $Min\ T = 156.162$. The specific route arrangement of each collecting vehicle has been listed above.

V. Conclusion and Prospects for Future Research

This paper firstly summarizes the research history and application scenarios of PSO, and proposes a **DSCPSO** algorithm combining parameter descent, SAA and Cauchy mutation according to the actual defects of the standard PSO. Secondly, the algorithm is given statistical and measure theory parameter analysis, variance analysis and convergence analysis, and finally applied the algorithm to the actual problem for the scheduling problem of specimen collection vehicles. Besides, in the processing of practical problems, we can also see that the convergence speed of **DSCPSO** is still not very fast, and it is greatly affected by the parameter settings, so that it cannot converge to the global best after enough iterations under the values of some parameters. Therefore, in the future, it is still necessary to explore better ways to improve the algorithm so that the algorithm converges faster and is less affected by parameter settings.

References

[1] Kennedy. J , Eberhart. R , Particle Swarm Optimization// Icnn95-international Conference on Neural Networks. IEEE, 1995.

[2] K.E Parsopolus, V.P Plagianakos, M.N Vrahatis, Objective Funcion “streching” to Alleviate Convergence to Local Minima, Nonlinear Analysis TMA47, 3419-3424, 2001.

[3]Anna Rakitianskaia, Andries Engelbrecht, Saturation in PSO Network Training, University of Pretoria, Department of Computer Science, Pretoria, South Africa, 2016.

[4]Zhang Lei, Pattern Classification Algorithm Based on Parallel PSO and Its Application Research，Jinan University, Department of Computer Sciencce, Jinan, China, 2005.

[5]Colorni. A. Dorigo. M. Maniezzo V. An investigation of some propertiesof an ant algorithm[C].In:Proc.Of parallel Problem Solving from Nature(PPSN) France:Elsiver 192.

[6]X.hu, Y.Shi and R.Eberhart, Recentt Advenes in Particle Swarm, Congress on Evolutionary Computation, Portland, Oregon, June 19-23,90-97,2004.

[7]Y.Shi and R.Eberhart, A Modified Particle Swarm Optimizer, Department of Electrical Engineering, Indiana University Purdue University Indianapolis, IN 46202-5160.

[8]He Jia, Wu Yaowu, Lou Suhua, Reactive power optimization of power system based on SA-PSO. Acta Power System and Its Automation, 2007, 19(5):5.

[9]Gong Zhaohua. Identification of kinetic parameters of microbial batch fermentation based on PSO algorithm. Computer Engineering and Applications, 2007, 043(027):p193-p195.

[10]Yang Yingjie, Particle Swarm Optimization and its Applivation, Beijing: Beijing

Institute of Technology Press, 2017: p11-p13.
[11]Li Ning, Zou Tong, Sun Debao. Research on Particle Swarm Optimization for Vehicle Routing Problem. Journal of Systems Engineering, 2004, 19(6):5.
[12]Qi Xuemei, Luo Yonglong, Zhao Cheng. Hybrid particle swarm algorithm for solving flow shop scheduling problem. Computer Engineering and Applications, 2012, 48(9):33-36.
[13]Chen Guimin, Jia Jianyuan, Han Qi, Study on the Stategy of Decreasing Inertia Weight in Particle Swarm Optimization Algorithm, School of Electro-Mechanical Engineering, Xi'an University, Xi'an, China, 710071, 2006.
[14]Langdon W.B, Poli.R, Evoluing Problems to Learn about Particle Swarm and Other Optimizers, in:Proc, CEC-2005, vol.1, 2005, p81-88.
[15]Hui Wang, Yong Liu, Changhe Li, Sanyou Zeng, Ahybrid particle swarm algorithm with Cauchy Mutation, IEEE Swarm Intelligence Symposimu 2007 (SIS 2007), Honolulu Hawaii, USA 2006, in press.
[16]Hui Wang, Yong Liu, Changhe Li, Sanyou Zeng, Opposition-based Particle Swarm Algorithm with Cauchy Mutation, China University of Geoscience, School of Computer Science, Wuhan China, 2014.
[17]Xu Haotian, Research on Improvement and Application of Multidimensional Optimization Based on Particle Swarm Optimization, Harbin Normal University, Harbin China, May 2021.
[18]Wang Yongji, Su Tingting, Liu Lei, Multi-strategy Cooperative Evolutionary PSO Based on Cauchy Mutation Strategy, National Key Laboratory of Science and Technology on Multispectual Information Processing, Automation College, Huazhong University of Science and Technology, Wuhan China, 430074, 2018.
[19]dos Santos Coelho. L, Herrera BM. Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system [J]. IEEE Trans. Ind. Electron. 2007, 54(6):3234-3245.
[20]Eberhart. R, Y. Shi, Particle swarm optimization: Developments, applications and resources. in: Proc. IEEE Congr. Evol. Comput. 2001, 1(1), 81-86
[21]Clerc. M. Initialisations for particle swarm optimization, 2008.
[22]Wang Wanliang, Tang Yu, The state of art in particle swarm optimization algorithms, College of Infomation Engineering, Zhejiang University of Technology, Hangzhou, 310032, China, 2007.
[23]Yang Yingjie, Particle Swarm Optimization and its Applivation, Beijing: Beijing Institute of Technology Press, 2017: 76-77.
[24]Mao Shisong, Cheng Yiming, Pu Xiaolong, Probability Theory and Mathematical Statistics Course, Beijing: Higher Education Press, 2019, 376-377,379-380
[25]Zeng Chaojian, Jie Jing, Cui Zhihua, Particle Swarm Optimiztion, Beijing:

Science Press, 2004:61-69

[26]国家卫健委：我国新冠病毒核酸检测能力达单管每日5700万管

[27]http://www.chenghua.gov.cn/chqrmzfw/c143868/chgk_zrdl.shtml

[28]Yue Xiaopemg, Quan Qixun, He Yuehua, Optimization problem of public bicycle scheduling based on particle swarm optimization，Technology Innovation and Application, 461000, Xuchang, China, 2022.

Acknowledgement

The topic of this paper was inspired by thinking about the optimization of classroom seating, so I started to learn about optimization algorithms, trying to find ways to improve traditional optimization algorithms by myself and applied what I learned to a broader and more interesting problem - nucleic acid detection optimization.

Through this study, I have benefited a lot, not only understand the operation principle of the optimization algorithm, but also understand the operation mode of my country's nucleic acid detection system. However, I can't study successfully without the help of many people around me. First of all, I tend to express my sincere gratitude to my math teacher Mr. Chen Xuefeng who instructs me to write and revise this paper and gives his passionate encouragement through the entire work. Besides, I want to thank Mr. Zhong Hao from Chengdu University of Technology that gives me rigorous mathematical theory guidance. Last but not least, I am very grateful for the support of my family, my head teacher Mr. Yao Meng and my school Chengdu Shude Middle School (Guanghua Campus). Thank you for your understanding, support and unreserved encouragement in the tense study situation of senior year and I will not finish this hard work without you all.

Appendix

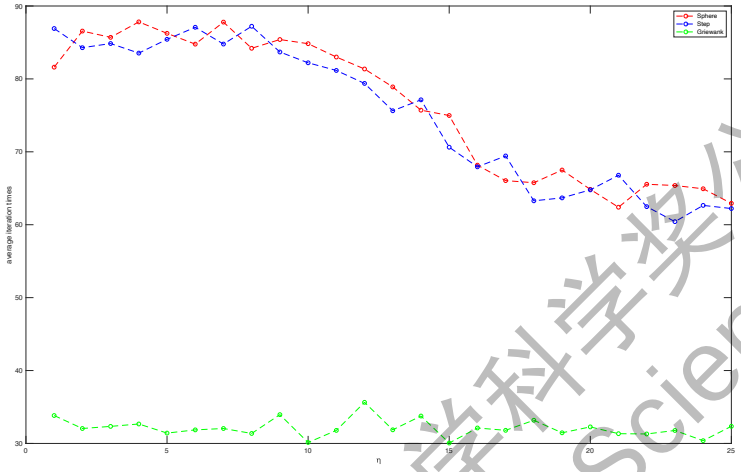
1

η	ITEM	*SPHERE*	*STEP*	*GRIEWANK*	*RASTRIGRIN*
$\eta = 0.5$	average iteration times	82.27	85.73	32.42	16.2010
	variance when converge below to 0.001 or 0.1	0.063148	0.0139	0.0526	/
	variance after 1000(200) iteration times	7.8777×10^{-90}	1.4425×10^{-10}	1.1824×10^{-3}	0.9975
$\eta = 1.0$	average iteration times	84.25	86.53	32.40	15.512
	variance when converge below to 0.001 or 0.1	3.0192	38.2394	0.0683	/
	variance after 1000(200) iteration times	2.8341×10^{-84}	3.0516×10^{-9}	1.6673×10^{-3}	4.7283
$\eta = 1.5$	average iteration times	85.49	86.49	32.62	15.0878
	variance when converge below to 0.001 or 0.1	14.9765	316.2976	0.0659	/

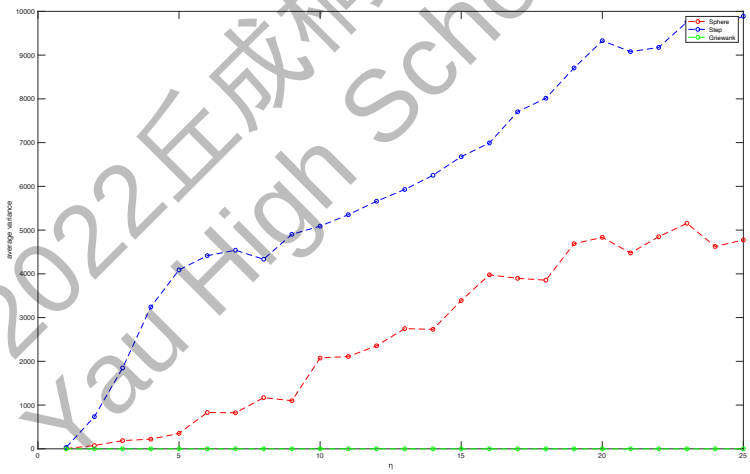
	variance after 1000(200) iteration times	2.1556×10^{-84}	4.2282×10^{-9}	5.1482×10^{-3}	11.8033
$\eta = 2.0$	average iteration times	86.93	87.04	32.51	15.1452
	variance when converge below to 0.001 or 0.1	37.7928	816.5862	0.0714	/
	variance after 1000(200) iteration times	3.2382×10^{-67}	4.2101×10^{-8}	0.1095	19.0258
$\eta = 2.5$	average iteration times	85.78	86.04	32.16	15.3778
	variance when converge below to 0.001 or 0.1	79.3801	1456.895	0.0750	/
	variance after 1000(200) iteration times	4.4783×10^{-74}	7.4567×10^{-8}	8.9659×10^{-3}	31.9342
$\eta = 3.0$	average iteration times	85.80	86.09	32.11	14.9362
	variance when converge below to 0.001 or 0.1	124.9039	1873.732	0.0730	/

variance	2.3293	7.5383	7.0057×10^{-3}	51.5823
after 1000(200)	$\times 10^{-79}$	$\times 10^{-8}$		
iteration times				

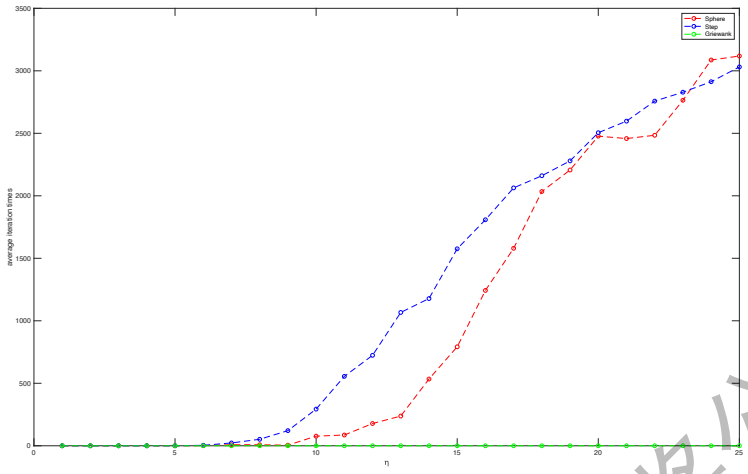
Table(1-1) the running conditions of various values of η(0.5~3.0)



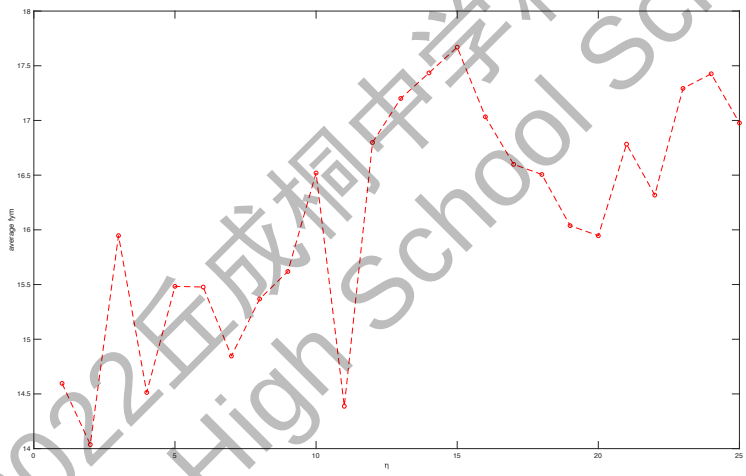
Figure(1-2) the average iteration times of η(1~25)

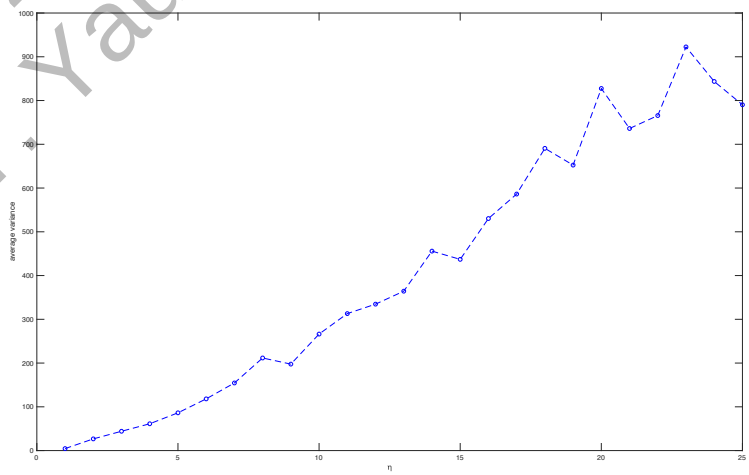


Figure(1-3) the average variances when converge below to 0.001 or 0.1 of η(1~25)

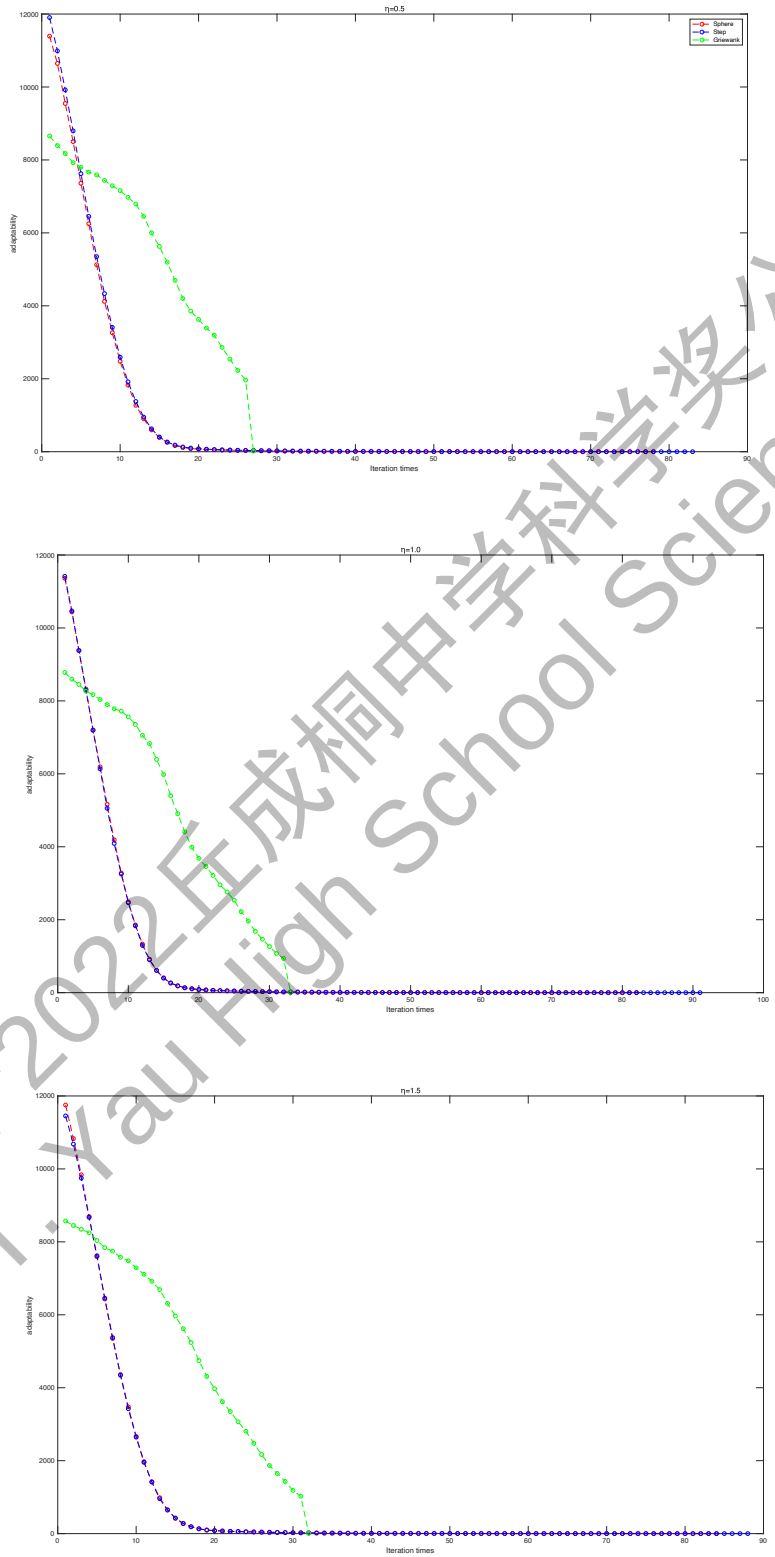


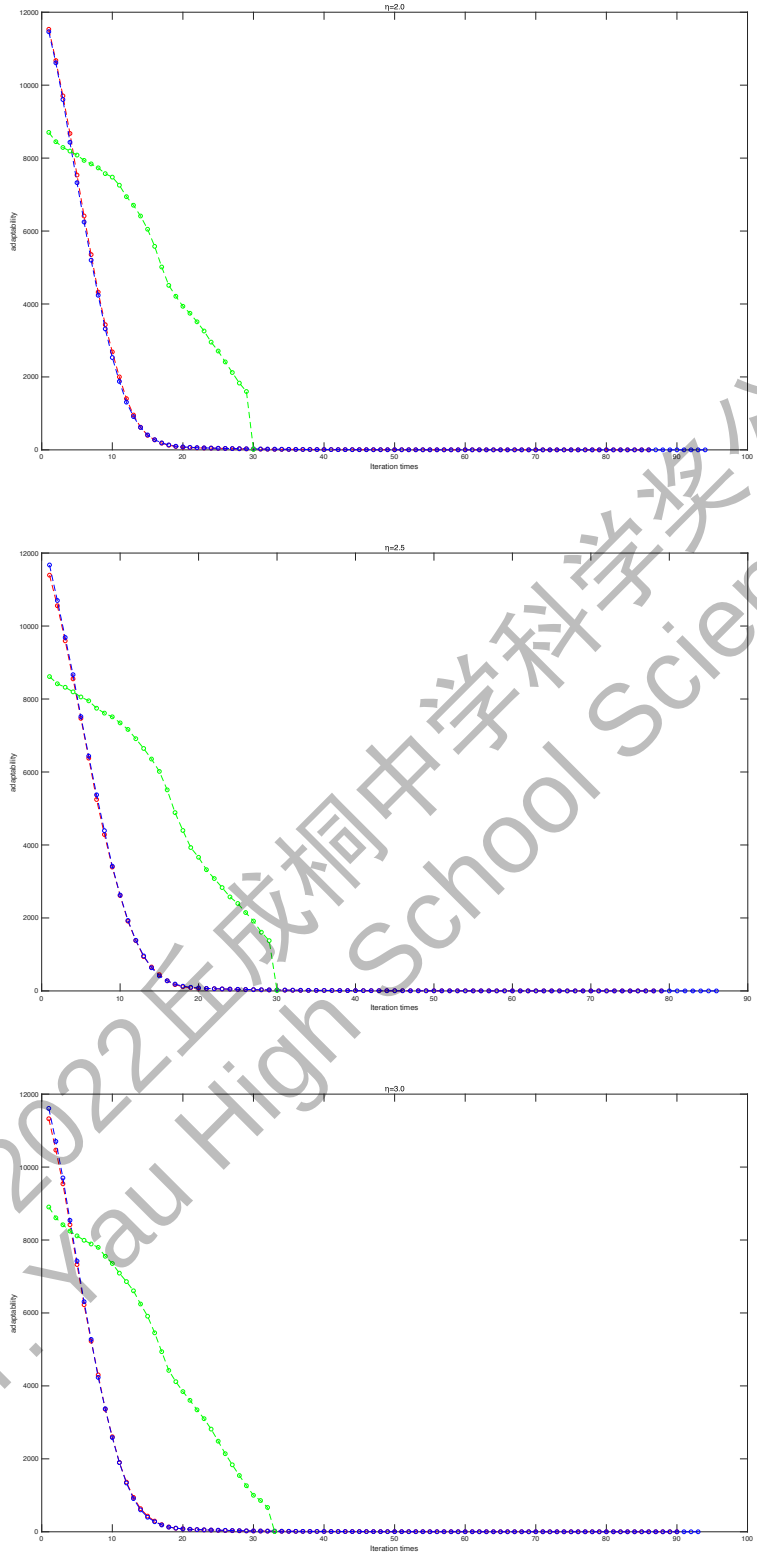
Figure(1-4) the average variances after 1000 iteration times of $\eta(1\sim25)$



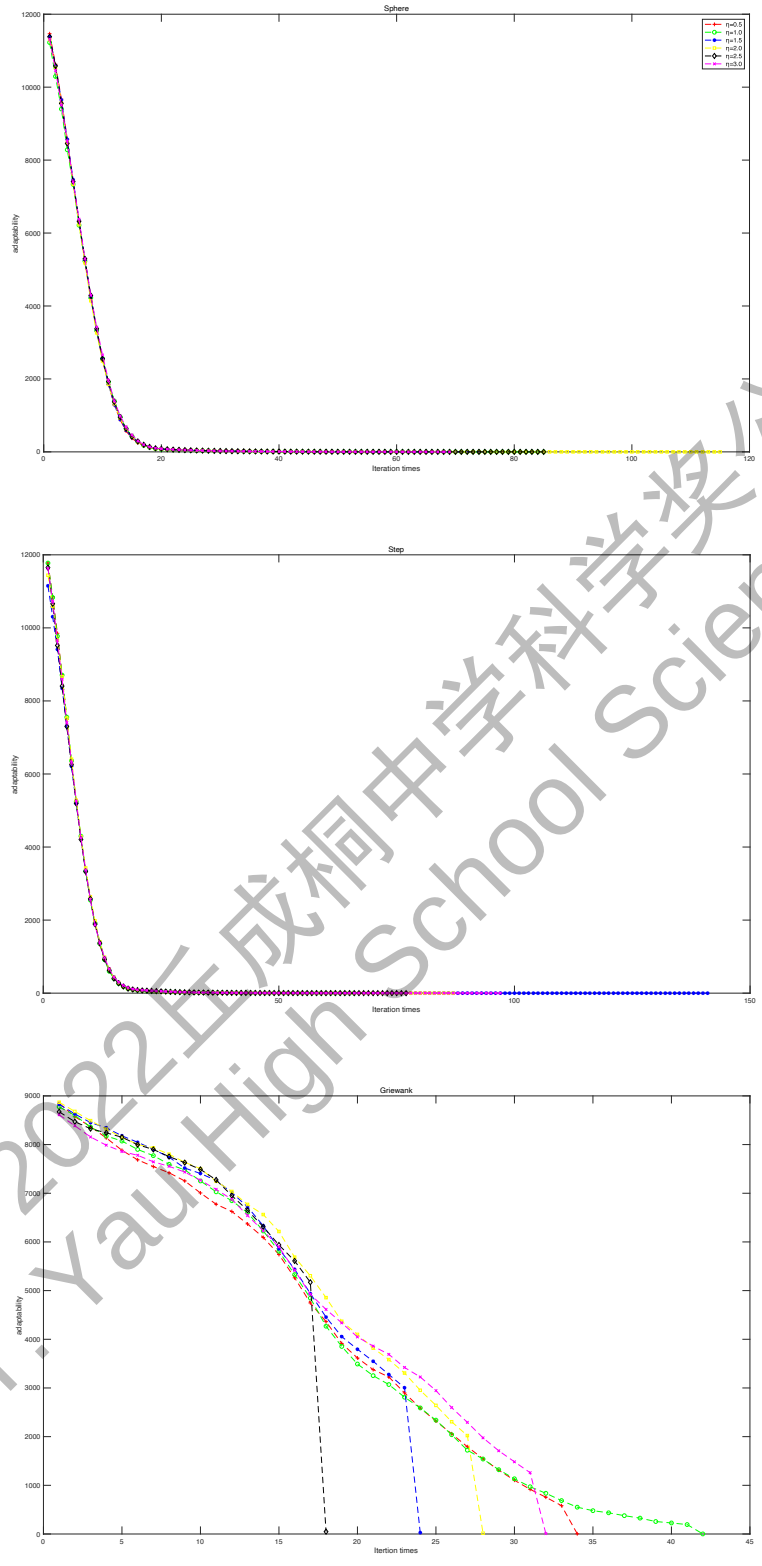


Figure(1-5) the average minimum value and average variances after 200 iteration times for *Rastrigrin* of $\eta(1\sim25)$





Figure(1-6), the convergence speed of various values of η



Figure(1-7) the convergence of 3 test functions in various values of η

2-1-2

	0.3	0.6	0.9	1.2	1.5	1.8
1	0.0012949	1.6041 $\times 10^{-8}$	4.4033 $\times 10^{-4}$	0.062499	0.14601	0.12472
2	2.417 $\times 10^{-4}$	1.1802 $\times 10^{-9}$	6.5264 $\times 10^{-4}$	0.057963	0.078264	0.11318
3	0.0053041	1.8755 $\times 10^{-9}$	2.0568 $\times 10^{-4}$	0.086992	0.080173	0.19478
4	5.1901 $\times 10^{-4}$	2.277 $\times 10^{-7}$	5.6277 $\times 10^{-4}$	0.084089	0.088556	0.085695
5	0.0016046	6.456 $\times 10^{-10}$	4.0918 $\times 10^{-4}$	0.064591	0.10793	0.16543
SAMPLE SUM	0.00896431	2.4751 $\times 10^{-7}$	0.0022706	0.356061	0.501833	0.683805
SAMPLE MEAN	0.00179286	4.9503 $\times 10^{-8}$	4.5412 $\times 10^{-4}$	0.0712122	0.100367	0.136761

Table(2-1-2-1') the running result of *program** for *Griewank*

2-2-1

	0.3	0.6	0.9	1.2	1.5	1.8
1	1.7772 $\times 10^{-4}$	5.3164 $\times 10^{-5}$	6.4696 $\times 10^{-5}$	1.2224 $\times 10^{-4}$	2.5530 $\times 10^{-4}$	0.0016958
2	1.8628 $\times 10^{-4}$	1.0222 $\times 10^{-4}$	2.0683 $\times 10^{-4}$	1.3756 $\times 10^{-4}$	2.8332 $\times 10^{-4}$	0.0018950
3	1.2334 $\times 10^{-4}$	1.6922 $\times 10^{-5}$	7.5491 $\times 10^{-5}$	0.0010078	7.6053 $\times 10^{-4}$	0.0014155

4	9.0429 $\times 10^{-5}$	1.5162 $\times 10^{-5}$	5.7082 $\times 10^{-4}$	1.1975 $\times 10^{-4}$	3.2181 $\times 10^{-4}$	0.0034476
5	2.9070 $\times 10^{-5}$	7.5964 $\times 10^{-5}$	1.3679 $\times 10^{-4}$	1.5019 $\times 10^{-4}$	9.0916 $\times 10^{-4}$	0.0014146
SAMP LE SUM	8.6846 $\times 10^{-4}$	1.9506 $\times 10^{-4}$	0.0010554(	0.00153754	0.00253462	0.0098685
SAMP LE MEAN	1.7369 $\times 10^{-4}$	3.9010 $\times 10^{-5}$	2.1109 $\times 10^{-4}$	3.0751 $\times 10^{-4}$	5.0692 $\times 10^{-4}$	0.0019737

Table(2-2-1-1) the running result of *program** for *Sphere*

2-2-2

	0.3	**0.6**	**0.9**	**1.2**	**1.5**	**1.8**
1	4.3721 $\times 10^{-6}$	1.2300 $\times 10^{-5}$	4.1752 $\times 10^{-5}$	4.6944 $\times 10^{-5}$	1.2267 $\times 10^{-4}$	1.0669 $\times 10^{-4}$
2	1.7415 $\times 10^{-5}$	1.8235 $\times 10^{-5}$	2.5218 $\times 10^{-5}$	9.3278 $\times 10^{-5}$	8.9004 $\times 10^{-5}$	1.0098 $\times 10^{-4}$
3	2.2207 $\times 10^{-5}$	1.8470 $\times 10^{-5}$	2.1060 $\times 10^{-5}$	3.0316 $\times 10^{-5}$	1.4308 $\times 10^{-4}$	3.6658 $\times 10^{-4}$
4	6.7572 $\times 10^{-6}$	1.6948 $\times 10^{-5}$	7.3658 $\times 10^{-5}$	1.6191 $\times 10^{-5}$	1.2259 $\times 10^{-4}$	3.2529 $\times 10^{-4}$
5	1.3753 $\times 10^{-5}$	4.7548 $\times 10^{-5}$	1.4000 $\times 10^{-5}$	3.5994 $\times 10^{-5}$	3.3163 $\times 10^{-4}$	1.3219 $\times 10^{-4}$
SAMPLE SUM	6.4504 $\times 10^{-5}$	1.1350 $\times 10^{-4}$	1.7569 $\times 10^{-4}$	2.2272 $\times 10^{-4}$	8.0897 $\times 10^{-4}$	0.0010318
SAMPLE MEAN	1.2901 $\times 10^{-5}$	2.2700 $\times 10^{-5}$	3.5138 $\times 10^{-5}$	4.4545 $\times 10^{-5}$	1.6180 $\times 10^{-4}$	2.0637 $\times 10^{-4}$

Table(2-2-2-1') the running result of *program** for *Griewank*

2-3-1

	0.3	0.6	0.9	1.2	1.5	1.8
1	1.006 $\times 10^{-4}$	1.8634 $\times 10^{-5}$	3.4151 $\times 10^{-5}$	6.3917 $\times 10^{-5}$	2.7404 $\times 10^{-4}$	0.0022169
2	8.9907 $\times 10^{-5}$	1.3518 $\times 10^{-4}$	4.7313 $\times 10^{-5}$	3.6628 $\times 10^{-4}$	0.002732	0.0033504
3	4.4579 $\times 10^{-5}$	2.7222 $\times 10^{-5}$	6.7139 $\times 10^{-5}$	4.2228 $\times 10^{-4}$	0.0011308	0.0096311
4	3.9816 $\times 10^{-5}$	8.0877 $\times 10^{-5}$	6.9826 $\times 10^{-5}$	3.3470 $\times 10^{-4}$	6.8168 $\times 10^{-4}$	0.0017696
5	3.6806 $\times 10^{-5}$	3.2860 $\times 10^{-4}$	1.6092 $\times 10^{-4}$	2.2928 $\times 10^{-4}$	0.0010177	0.0020468
SAMPLE SUM	3.1171 $\times 10^{-4}$	5.9051 $\times 10^{-4}$	2.3452 $\times 10^{-4}$	0.00141646	0.0058362	0.0190148
SAMPLE MEAN	6.2342 $\times 10^{-5}$	1.1810 $\times 10^{-4}$	4.6904 $\times 10^{-5}$	2.8329 $\times 10^{-4}$	0.0011672	0.0038029

Table(2-3-1-1) the running result of *program** for *Sphere*

2-3-2

	0.3	0.6	0.9	1.2	1.5	1.8
1	7.7216 $\times 10^{-6}$	1.6864 $\times 10^{-5}$	3.8373 $\times 10^{-5}$	1.0743 $\times 10^{-4}$	1.1636 $\times 10^{-4}$	9.2858 $\times 10^{-4}$
2	3.7280 $\times 10^{-5}$	8.2666 $\times 10^{-5}$	3.8869 $\times 10^{-5}$	8.4031 $\times 10^{-5}$	3.5272 $\times 10^{-4}$	3.9516 $\times 10^{-4}$
3	1.0579 $\times 10^{-5}$	5.3732 $\times 10^{-5}$	1.0804 $\times 10^{-4}$	3.2744 $\times 10^{-5}$	1.5675 $\times 10^{-4}$	0.001019

4	1.6088×10^{-5}	2.3969×10^{-5}	1.4721×10^{-5}	4.5268×10^{-5}	1.8329×10^{-4}	0.0010383
5	1.9421×10^{-5}	7.1276×10^{-6}	5.4692×10^{-5}	3.4366×10^{-5}	1.3393×10^{-4}	4.1852×10^{-4}
SAMPLE SUM	9.1090×10^{-5}	1.8436×10^{-4}	2.5470×10^{-4}	3.0384×10^{-4}	9.4305×10^{-4}	0.0037996
SAMPLE MEAN	1.8218×10^{-5}	3.6872×10^{-5}	5.0939×10^{-5}	6.0768×10^{-5}	1.8861×10^{-4}	7.5991×10^{-4}

Table(2-3-2-1') the running result of *program** for *Griewank*

2-4

	0.3	**0.6**	**0.9**	**1.2**	**1.5**	**1.8**	**SUM**	**MEA**
0.2	0.058815	3.9338×10^{-6}	3.0822×10^{-5}	1.8762	1.8065	1.8762	15.86424	0.881347
	0.040233	5.6933×10^{-6}	1.9204×10^{-4}	1.8024	2.0664	2.0661		
	0.01707	4.8852×10^{-7}	9.1280×10^{-5}	1.3138	1.4701	1.4701		
0.4	0.034997	7.7237×10^{-8}	1.0120×10^{-4}	1.2784	1.2477	2.4816	11.66321	0.64796
	0.028996	1.1219×10^{-7}	2.0222×10^{-5}	0.7744	0.89292	0.89292		
	0.066247	1.2838×10^{-8}	1.6910×10^{-4}	0.73024	1.5290	1.7055		
0.6	0.080422	2.2502×10^{-6}	7.5734×10^{-6}	1.1211	1.5944	2.7345	15.01484	0.83416
	0.051915	2.5630×10^{-5}	4.8091×10^{-5}	1.2561	1.7165	2.6119		

	0.041247	3.5050×10^{-7}	1.0594×10^{-5}	0.98958	0.73688	2.0802		
0.8	0.02092	5.3524×10^{-7}	1.1606×10^{-5}	0.064315	0.094316	1.9687	9.82748	0.54597
	0.15174	4.7816×10^{-10}	2.7555×10^{-5}	0.19917	1.5609	1.9007		
	0.054809	1.4669×10^{-5}	8.1372×10^{-5}	0.38297	1.4267	1.9841		
1.0	0.023028	1.7440×10^{-6}	1.9485×10^{-4}	0.37114	0.82969	1.9951	10.36774	0.57599
	0.027039	1.6064×10^{-5}	5.1985×10^{-4}	0.44487	0.96194	1.4944		
	0.009571	3.0003×10^{-6}	4.1863×10^{-5}	0.45858	1.0068	2.7448		
1.2	0.045529	1.3392×10^{-6}	1.3942×10^{-4}	0.28592	0.99249	1.6802	10.13084	0.56282
	0.011814	1.0244×10^{-7}	1.6050×10^{-4}	0.41934	0.64282	2.5300		
	0.02788	5.1773×10^{-6}	1.9676×10^{-4}	0.57929	0.90615	2.0089		
SUM	0.792272	8.1177×10^{-5}	0.002044	14.34782	21.48221	36.24392		
MEAN	0.044015	4.50891×10^{-6}	1.1359×10^{-4}	0.797101	1.193456	2.013551		

Table(2-4) the running result of *program** (two-way ANOVA) for *Sphere*

3-1

$\widehat{\mu_A} = 0.611362362195$, $\widehat{\mu_{A'}} = 0.05176453825$, $\widehat{\mu_B} = 5.3532 \times 10^{-4}$, $\widehat{\mu_{B'}} = 8.0576 \times 10^{-5}$, $\widehat{\mu_C} = 0.00913456$, $\widehat{\mu_{C'}} = 1.8589 \times 10^{-4}$

$\widehat{a_{A1}} = -0.580143282195$, $\widehat{a_{A2}} = -0.611362029025$, $\widehat{a_{A3}} = -0.606857602195$, $\widehat{a_{A4}} = 0.006387637805$, $\widehat{a_{A5}} = 0.762937637805$, $\widehat{a_{A6}} = 1.065037637805$; $\widehat{a_{A'1}} = -0.04997167825$, $\widehat{a_{A'2}} = -0.051764488747$, $\widehat{a_{A'3}} = -0.05131041825$, $\widehat{a_{A'4}} = 0.01944766175$, $\widehat{a_{A'5}} = 0.04860246175$, $\widehat{a_{A'6}} = 0.08499646175$;

$\widehat{a_{B1}} = -3.6163 \times 10^{-4}$, $\widehat{a_{B2}} = -4.9631 \times 10^{-4}$, $\widehat{a_{B3}} = -3.2423 \times 10^{-4}$, $\widehat{a_{B4}} = 2.2781 \times 10^{-4}$, $\widehat{a_{B5}} = 2.8400 \times 10^{-5}$, $\widehat{a_{B6}} = 0.00143838$; $\widehat{a_{B'1}} = 6.7675 \times 10^{-5}$, $\widehat{a_{B'2}} = 5.7876 \times 10^{-5}$, $\widehat{a_{B'3}} = 4.5438 \times 10^{-5}$, $\widehat{a_{B'4}} = 3.6031 \times 10^{-4}$, $\widehat{a_{B'5}} = 8.1224 \times 10^{-5}$, $\widehat{a_{B'6}} = 1.2579 \times 10^{-4}$;

$\widehat{a_{C1}} = -8.5111 \times 10^{-4}$, $\widehat{a_{C2}} = -7.9536 \times 10^{-4}$, $\widehat{a_{C3}} = -8.6655 \times 10^{-4}$, $\widehat{a_{C4}} = 6.3017 \times 10^{-4}$, $\widehat{a_{C5}} = 2.5374 \times 10^{-4}$, $\widehat{a_{C6}} = 0.0028894$; $\widehat{a_{C'1}} = -1.6767 \times 10^{-4}$, $\widehat{a_{C'2}} = -1.4902 \times 10^{-4}$, $\widehat{a_{C'3}} = -1.3495 \times 10^{-4}$, $\widehat{a_{C'4}} = -1.2512 \times 10^{-4}$, $\widehat{a_{C'5}} = 2.7200 \times 10^{-6}$, $\widehat{a_{C'6}} = 5.7402 \times 10^{-4}$.

$\widehat{\sigma_A}^2 = 0.062792$, $\widehat{\sigma_{A'}}^2 = 0.014309$;

$\widehat{\sigma_B}^2 = 1.69535 \times 10^{-5}$, $\widehat{\sigma_{B'}}^2 = 4.6109 \times 10^{-7}$;

$\widehat{\sigma_C}^2 = 2.0833 \times 10^{-4}$, $\widehat{\sigma_{C'}}^2 = 1.9689 \times 10^{-6}$.

3-2

Set $\alpha = 0.1$, $t_{1-\frac{\alpha}{2}}(f_e) = t_{0.95}(24) = 1.7109$

$$\mu_A: \left[0.6113623 \pm 1.7109 \times \frac{\sqrt{0.062792}}{\sqrt{5}}\right] = [0.4992982, 0.7234266]$$

$$\mu_{A'}: \left[0.0517645 \pm 1.7109 \times \frac{\sqrt{0.014309}}{\sqrt{5}}\right] = [0, 0.14903245]$$

$$\mu_B: \left[5.3532 \times 10^{-4} \pm 1.7109 \times \frac{\sqrt{1.69535 \times 10^{-5}}}{\sqrt{5}}\right] = [0, 0.003685746]$$

$$\mu_{B''}: \left[8.0576 \times 10^{-5} \pm 1.7109 \times \frac{\sqrt{4.6109 \times 10^{-7}}}{\sqrt{5}}\right] = [0, 6.0013 \times 10^{-4}]$$

$$\mu_C: \left[0.00913456 \pm 1.7109 \times \frac{\sqrt{2.0833 \times 10^{-4}}}{\sqrt{5}}\right] = [0.0089751589, 0.0092939611]$$

$$\mu_{C'}: \left[1.8589 \times 10^{-4} \pm 1.7109 \times \frac{\sqrt{1.9689 \times 10^{-6}}}{\sqrt{5}}\right] = [0, 0.0059512125]$$

4

Number	Location	Coordinate
1	Mengzhuiwan	104.10236E , 30.67693N
2	Shuangqiaozi	104.11259E , 30.65368N
3	Bailianchi	104.14495E , 30.73159N
4	Jianshe Road	104.07275E , 30.57899N
5	Taoi Road	104.12218E, 30.68805N
6	Fuqing Road	104.12220E, 30.68806N
7	Erxian Bridge	104.13714E, 30.68958N
8	Tiaodeng River	104.12599E, 30.66222N
9	Wannianchang	104.14106E, 30.65184N
10	Shuangshuinian	104.08005E, 30.71514N
11	Shengdeng	104.11191E, 30.68805N
12	Baohe	104.15237E, 30.66407N
13	Qinglong (1)	104.11969E, 30.70961N
14	Qinglong (2)	104.11206E, 30.70761N
15	Longtan (1)	104.17110E , 30.71094N
16	Longtan (2)	104.17906E , 30.71167N
(17)	Sichuan Jinyu Medical Laboratory Center	104.17518E , 30.69329N
(18)	Chengdu Sixth People's Hospital	104.11468E , 30.67275N
(19)	Chengdu Chenghua District Center for Disease Control and Prevention	104.11639E , 30.71449N

	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	#14	#15	#16	#17	#18	#19
#1	0	3.4	11	2.3	3.5	3.4	4.7	4	5.7	6	6.2	7.7	5.3	4.7	13	14	11	1.7	6.7
#2	3.4	0	13	3.1	5.2	4.7	5.9	1.9	2.8	11	5.6	4.7	8.1	7	11	12	9.7	2.2	9.2
#3	11	13	0	11	8.4	8.4	6.9	11	14	9.4	6.1	13	4.3	5.2	7.3	7.7	8.1	11	4.3
#4	2.3	3.1	11	0	1.5	1.5	2.7	2.7	4.3	6.5	4.2	5.1	5.3	4.2	9.1	9.2	8.6	1.3	6.2
#5	3.5	5.2	8.4	1.5	0	0.15	2	3.9	4.7	6.7	3.5	5.8	3.9	4.3	8.4	8.2	8.6	2.5	5.5
#6	3.4	4.7	8.4	1.5	0.15	0	2.1	4	4.8	6.5	3.6	5.7	4	4.2	8.4	8.1	9	2.6	5.6
#7	4.7	5.9	6.9	2.7	2	2.1	0	5.8	6.4	8.5	1.7	5.9	4.5	4.6	6	6.6	5.2	4.6	4.9
#8	4	1.9	11	2.7	3.9	4	5.8	0	4	8.9	5.6	5.4	8.1	6.9	10	11	8.6	2.6	8.7
#9	5.7	2.8	14	4.3	4.7	4.8	6.4	4	0	11.3	5.6	2.5	9.7	8.8	9.6	10.4	8.4	4.8	12
#10	6	11	9.4	6.5	6.7	6.5	8.5	8.9	11.3	0	12.1	15.3	5	4.4	13.7	14.7	14	8.4	4.9
#11	6.2	5.6	6.1	4.2	3.5	3.6	1.7	5.6	5.6	12.1	0	5.1	4.7	5.7	4.2	4.8	3.9	4.9	5.5
#12	7.7	4.7	13	5.1	5.8	5.7	5.9	5.4	2.5	15.3	5.1	0	8.1	8.4	8.5	9	7.8	5.3	9.3
#13	5.3	8.1	4.3	5.3	3.9	4	4.5	8.1	9.7	5	4.7	8.1	0	1.2	8.2	9.2	8.1	6.3	1.1
#14	4.7	7	5.2	4.2	4.3	4.2	4.6	6.9	8.8	4.4	5.7	8.4	1.2	0	8.7	9.5	9.5	5.2	2.5
#15	13	11	7.3	9.1	8.4	8.4	6	10	9.6	13.7	4.2	8.5	8.2	8.7	0	1.5	3.7	9.4	8.2
#16	14	12	7.7	9.2	8.2	8.1	6.6	11	10.4	14.7	4.8	9	9.2	9.5	1.5	0	3.2	11.7	9
#17	11	9.7	8.1	8.6	8.6	9	5.2	8.6	8.4	14	3.9	7.8	8.1	9.5	3.7	3.2	0	9.9	9.1
#18	1.7	2.2	11	1.3	2.5	2.6	4.6	2.6	4.8	8.4	4.9	5.3	6.3	5.2	9.4	11.7	9.9	0	7.2
#19	6.7	9.2	4.3	6.2	5.5	5.6	4.9	8.7	12	4.9	5.5	9.3	1.1	2.5	8.2	9	9.1	7.2	0

Program*（This example is for the inertia factor (ω) under *Sphere*）

```
clear;
clc;
N=100;
D=10;
ger=100;
c1=1.5;
c2=1.5;
w1=0.3;
w2=0.6;
w3=0.9;
w4=1.2;
w5=1.5;
w6=1.8;
eps=0.001;

x=zeros(N,D);
v=zeros(N,D);
for i=1:N
    for j=1:D
        x(i,j)=randn;
        v(i,j)=randn;
    end
end

figure(1);
for j=1:D
```

```
    if (rem(D,2)>0)
        subplot((D+1)/2,2,j);
    else
        subplot(D/2,2,j);
    end
    plot(x(:,j),'b*');
    grid on;
    xlabel('The particles');
    ylabel('initial position');
    tInfo=strcat(char(j+48),'dimension');
    if (j>9)

tInfo=strcat(char(floor(j/10)+48),char(rem(j,10)+48),'dimension');
    end
    title(tInfo);
end

figure(2);
for j=1:D
    if (rem(D,2)>0)
        subplot((D+1)/2,2,j);
    else
        subplot(D/2,2,j);
    end
    plot(v(:,j),'b*');
    grid on;
    xlabel('The particles');
    ylabel('initial velocity');
    tInfo=strcat(char(j+48),'dimension');
    if (j>9)

tInfo=strcat(char(floor(j/10)+48),char(rem(j,10)+48),'dimension');
    end
    title(tInfo);
end
figure(3);
subplot(1,2,1);

x1=x;
v1=v;

p1=x1;
pbest1=ones(N,1);
for i=1:N
```

```
    pbest1(i)=fitness(x1(i,:),D);
end

g1=1000*ones(1,D);
gbest1=1000;
for i=1:N
    if (pbest1(i)<gbest1)
        g1=p1(i,:);
        gbest1=pbest1(i);
    end
end
gb1=ones(1,ger);

for i=1:ger
    for j=1:N
        if (fitness(x1(j,:),D)<pbest1(j))
            p1(j,:)=x1(j,:);
            pbest1(j)=fitness(x1(j,:),D);
        end
        if (pbest1(j)<gbest1)
            g1=p1(j,:);
            gbest1=pbest1(j);
        end
        v1(j,:)=w1*v1(j,:)+c1*rand*(p1(j,:)-x1(j,:))+c2*rand*(g1-
x1(j,:));
        x1(j,:)=x1(j,:)+v1(j,:);
    end
    gb1(i)=gbest1;
end
plot(gb1);
disp(['gb1:',num2str(gbest1)]);
TempStr=sprintf('w=%g',w1);
title(TempStr);
xlabel('Iteration times');
ylabel('Fitness value');

%% function
function result=fitness(x,D)
sum=0;
for i=1:D
    sum=sum+x(i)^2;
end
result=sum;
end
```

Program of Section IV

```
clear
clc
close all
clear global
global para_v para_h para_X totle_num para_D
para_D=[0 3.4 11 2.3 3.5 3.4 4.7 4 5.7 6 6.2 7.7 5.3 4.7 13 14 11 1.7
6.7
       3.4 0 13 3.1 5.2 4.7 5.9 1.9 2.8 11 5.6 4.7 8.1 7 11 12 9.7
2.2 9.2
       11 13 0 11 8.4 8.4 6.9 11 14 9.4 6.1 13 4.3 5.2 7.3 7.7 8.1 11
4.3
       2.3 3.1 11 0 1.5 1.5 2.7 2.7 4.3 6.5 4.2 5.1 5.3 4.2 9.1 9.2
8.6 1.3 6.2
       3.5 5.2 8.4 1.5 0 0.15 2 3.9 4.7 6.7 3.5 5.8 3.9 4.3 8.4 8.2
8.6 2.5 5.5
       3.4 4.7 8.4 1.5 0.15 0 2.1 4 4.8 6.5 3.6 5.7 4 4.2 8.4 8.1 9
2.6 5.6
       4.7 5.9 6.9 2.7 2 2.1 0 5.8 6.4 8.5 1.7 5.9 4.5 4.6 6 6.6 5.2
4.6 4.9
       4 1.9 11 2.7 3.9 4 5.8 0 4 8.9 5.6 5.4 8.1 6.9 10 11 8.6 2.6
8.7
       5.7 2.8 14 4.3 4.7 4.8 6.4 4 0 11.3 5.6 2.5 9.7 8.8 9.6 10.4
8.4 4.8 12
       6 11 9.4 6.5 6.7 6.5 8.5 8.9 11.3 0 12.1 15.3 5 4.4 13.7 14.7
14 8.4 4.9
       6.2 5.6 6.1 4.2 3.5 3.6 1.7 5.6 5.6 12.1 0 5.1 4.7 5.7 4.2 4.8
3.9 4.9 5.5
       7.7 4.7 13 5.1 5.8 5.7 5.9 5.4 2.5 15.3 5.1 0 8.1 8.4 8.5 9
7.8 5.3 9.3
       zeros(1,13) 1.2 8.2 9.2 8.1 6.3 1.1
       zeros(1,14) 8.7 9.5 9.5 5.2 2.5
       zeros(1,15) 1.5 3.7 9.4 8.2
       zeros(1,16) 3.2 11.7 9
       zeros(1,17) 9.9 9.1
       zeros(1,18) 7.2
       zeros(1,19)];
```

```
error_1=0;
error_index=[];
for i=1:size(para_D,1)
    for j=1:size(para_D,2)
        if para_D(i,j)==0&para_D(j,i)~=0
            para_D(i,j)=para_D(j,i);
        elseif para_D(i,j)~=0&para_D(j,i)==0
            para_D(j,i)=para_D(i,j);
        elseif para_D(i,j)~=para_D(j,i)
            error_1=error_1+1;
            error_index(end+1,:)=[i,j];
        end
    end
end
para_v=0.5*60;
para_h=3/60;
para_X=[0.98,1.05,1.02];
totle_num=length(para_D)-3;
sizepop=1000;
dim=totle_num*2;
ger=500;
xlimit_max=ones(1,dim)*3-0.00001;
xlimit_min=zeros(1,dim)+0.00001;

vlimit_max=0.4*(xlimit_max-xlimit_min);
vlimit_min =-1*vlimit_max;

w=0.7;
c1=1;
c2=0.1;

pop_x=zeros(dim,sizepop);
pop_v=zeros(dim,sizepop);
for i=1:dim
    for j=1:sizepop
        pop_x(i,j)=(xlimit_min(i)+(xlimit_max(i)-xlimit_min(i))*rand);
        pop_v(i,j)=(vlimit_min(i)+(vlimit_max(i)-vlimit_min(i))*rand);
    end
end
gbest=pop_x;

for j=1:sizepop
          fitness_gbest(j)=fun_1(pop_x(:,j));
end
```

```
zbest=pop_x(:,1);
fitness_zbest=fitness_gbest(1);
for j=1:sizepop
    if fitness_gbest(j)<fitness_zbest
        zbest=pop_x(:,j);
        fitness_zbest=fitness_gbest(j);
    end
end
iter=1;

record_2=zeros(ger,1);

T=2000;
C_S=fitness_gbest;
d_t=0.98;
while iter<=ger
     c1=1-0.6*(iter/ger);
    fai=c1+c2;
    w=2/abs(2-fai-(fai^2-4*fai)/2);
    if iter>2
        C_S=fitness_pop;
    end
    for j=1:sizepop
        pop_v(:,j)=(w*pop_v(:,j)+c1*rand*(gbest(:,j)-
pop_x(:,j))+c2*rand*(zbest-pop_x(:,j)));

           for i=1:dim
          if  pop_v(i,j)>vlimit_max(i)
              pop_v(i,j)=vlimit_max(i);
          end
          if  pop_v(i,j)<vlimit_min(i)
              pop_v(i,j)=vlimit_min(i);
          end
        end
        pop_x(:,j)=pop_x(:,j)+pop_v(:,j);

             for i=1:dim
          if pop_x(i,j)>xlimit_max(i)
              pop_x(i,j)=xlimit_max(i);
          end
          if pop_x(i,j)<xlimit_min(i)
              pop_x(i,j)=xlimit_min(i);
          end
```

```
        end
             if rand>0.75
            pop_x(:,j)=(xlimit_min'+(xlimit_max'-
xlimit_min').*rand(dim,1));
        end

            fitness_pop(j)=fun_1(pop_x(:,j));
              if fitness_pop(j)<fitness_gbest(j)
            gbest(:,j)=pop_x(:,j);
            fitness_gbest(j)=fitness_pop(j);
        end
        if fitness_gbest(j)<fitness_zbest
            zbest=gbest(:,j);
            fitness_zbest=fitness_gbest(j);
        end
          if iter>2&C_S(j)>fitness_gbest(j)
            p=exp(-(C_S(j)-fitness_gbest(j)))/(iter*T);
            if rand<(1-p)
                temp_1=randperm(length(zbest),1);
                pop_x(temp_1,j)=zbest(temp_1);
            end
        end
    end
    T=T*d_t;
    pop_x(:,1)=zbest+zbest.*(((ger-iter)/ger)+tan((rand(dim,1)-
1/2)*pi)+1);

    record_2(iter)=fitness_zbest;
    iter=iter+1;
    disp(iter)
    disp(fitness_zbest)
end
figure()
plot(record_2,'LineWidth',1.5)
xlabel('µü´ú´ÎÊý','FontSize',12,'FontWeight','bold')
ylabel('Ä¿±êÖµ','FontSize',12,'FontWeight','bold')
title('µü´úÇúÏß','FontSize',15,'FontWeight','bold')
x=zbest;
X=ceil(x(1:totle_num));
X_1=x(totle_num+1:end);

for i=1:3
    temp_index=find(X==i);
    if length(temp_index)==0
```

```
        car_way(i,1)=0;
    else
        temp_X=sortrows([X_1(temp_index) temp_index],1);
        car_way(i,1:length(temp_index))=temp_X(:,2)';
    end
end
record_t=[];
p_0=[17,18,19];
p_0_1=[17,18,19];
for i=1:3
    t=0;
    temp_index=car_way(i,:);
    temp_index(temp_index==0)=[];
    if length(temp_index)>0
        for j=1:length(temp_index)
            t=t+para_D(temp_index(j),p_0(i))/para_v;
            record_t(temp_index(j),1:3)=[i,t,t+para_h];
            t=t+para_h;
            p_0(i)=temp_index(j);
        end
    end
    t=t+para_D(p_0(i),p_0_1(i))/para_v;
    final_t(i)=t;
end
y=sum(final_t.*para_X);
```