# A Novel Optimization Algorithm for Adaptive Simplex Method with Application to High Dimensional Functions

ZuoYing LIU[1], XianWen LUO[2]

1 Southwest University Chongqing 402460, China, zuyingliu@aliyun.com
2 Southwest University Chongqing 402460, China, xianwenl@swu.edu.cn.

***Abstract* — Focusing on the simplex method's (SM) slow convergence and poor ability to solve problems of high dimensional functions, a novel Adaptive simplex method (ASM) based on analyzing search trace of SM is proposed. Deflection and symmetry operations are introduced to ASM. Deflection operations deflect each vertex for a certain distance to better direction guided by the latest search trace. Symmetry operations make each vertex symmetrize about the optimal vertex and better symmetry vertices substitute for poor vertices. The improved adaptive simplex method not only increases the way to change the optimization direction but leads the optimization direction closer to the steepest descent direction, thus it accelerates the convergence speed and enhances the ability of solving the problem of high dimensional functions.**

***Keywords – simplex, simplex method (SM), high-dimensional-functions, optimization algorithm***

## I. INTRODUCTION

Simplex algorithm (Simplex Method, SM) has been widely used as a direct optimization method without differentiation [1,2]. Spendly first proposed Regular Simplex Evolutionary Technique [3], Nelder improved Simplex Method into a practical local optimization method [4]. In order to solve the problem of constrained optimization, Box modifies it into complex method [5]. Simplex refers to a convex polytope with n+1 vertex in n-dimensional space En. Set Vj as n+1 vertices in n-dimensional space, j=0，1，…，n，and V1－V0，V2－V0，V3－V0，…， Vn－V0 linear independence, V0 is any vertex in n+1 vertices, then the n+1 vertices form a simplex, and the optimal vertex, the worst vertex and the relatively worst vertex are Vb，Vw，Vp respectively. Simplex method compares the adaptive value of each vertex, constantly replaces the relatively poor vertex with the optimal vertex through the operations of reflection, expansion, contraction and compression so that the whole simplex move and contract towards the extreme point until it converges to the required precision.

However, Simplex (Complex) algorithm converges slowly [6,7] and is completely ineffective in solving problems of high-dimensional-functions [8]. The reflection operation is the most important operation in the algorithm, thus many improved methods at present focus on the reflection operation. One of them is using dynamic expansion to test reflection coefficient [9], which needs to increase the amount of computation; Others take the worst vertices as the reflection vertices [10,11], but the effect is limited; and still others make use of other optimization methods to find the most suitable reflection coefficient[12], which increases the complexity of the algorithm and the amount of computation. Han Zhixin et al proposed the revolving direction search algorithm of the complex shape in which each vertex reflects on the optimal vertex [13]. The effect is improved but the convergence speed is not fast enough; Zhang Xiaohong et al proposed to first rotate the reflection direction to a certain angle, then search in an annular-super-spherical region so that increases the success rate [14]. This method mainly aims at the complex problem of function surface. The author proposes a new adaptive simplex method (Adaptive Simplex Method, ASM) under the guidance of simplex search trajectory due to the following reasons: ① It is difficult to solve the problem of high-dimensional-function and dimension disaster appears in solving the problem of high-dimensional-function ; ② The descending speed of simplex search direction is not fast enough and it is not the steepest descent direction; ③ The number of updated vertices in each time is very little and only one vertex is updated; ④ The searching speed and step size are fixed, which makes the improved convergence speed still slow and it is still very difficult to solve the problem of high-dimensional-function. Taking the minimum value as an example, the algorithm proposed in this paper accelerates the optimizing speed and enhances the ability of solving the problem of high-dimensional-function.

## II. IMPROVEMENT OF SIMPLEX

### *A. Introduction of the deflection operation*

Determine the current search direction according to the historical trajectory of simplex search, especially the recent change of simplex's migration direction, which can make the search closer to the descent direction and even the steepest descent direction. Define the migration directions of t、t-1generation as follows:

$$\vec{e}(0)=0;\vec{e}(t)=\frac{V_b(t)-V_w(t\text{-}1)}{\|V_b(t)-V_w(t\text{-}1)\|}(t\neq 0) \quad (1)$$

Then the optimal direction of t generation can be defined as:

$$\vec{E}(0)=0;\vec{E}(t)=\frac{(1-\rho)\vec{E}(t\text{-}1)+\rho\vec{e}(t)}{\|(1-\rho)\vec{E}(t\text{-}1)+\rho\vec{e}(t)\|}(t\neq 0) \quad (2)$$

In the formula, $V_b(t)$、$V_w(t)$ are the optimal and worst vertices of t generation simplex, $\rho\in[0,1]$ is the forgetting rate of historical optimizing direction. Then let each vertex deflect the length of rj to optimizing direction from the worst point Vw and get Yj：

$$Y_j=(V_j-V_w)+r_j\vec{E}(t),j=0,1,\cdots,n\wedge j\neq w \quad (3)$$

Set the vertex of the new simplex as Yj，j=0，1，…，n，if the first m Yj are better than the corresponding Vj, get the new simplex vertex from equation（3）:

$$Y_j=\begin{cases}(V_j-V_w)+r_j\vec{E}(t) & j=0,1,\cdots,m\text{-}1\\ V_j & j=m,m+1,\cdots,n\end{cases} \quad (4)$$

Only when Yj-Vw is not parallel to Yi-Vw（j=0，1，…，m-1；i=0，1，…，n，i≠j, the new simplex does not degrade. Then it comes to the value of deflection rj, set lj as n numbers which are not all 0, then

$$\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)+\sum_{j=0,j\neq w}^{n}l_j(r_j\vec{E}-V_w)$$

In the practical application, the deflection distance rj of each vertex is the same, that is, rj=r，j=0，1，…，n，so the above formula is simplified as:

$$=\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)+r\vec{E}\sum_{j=0,j\neq w}^{n}l_j-V_w\sum_{j=0,j\neq w}^{n}l_j$$

Set the above formula is not equal to 0, then

$$r\vec{E}\sum_{j=0,j\neq w}^{n}l_j\neq V_w\sum_{j=0,j\neq w}^{n}l_j-\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)\Leftrightarrow r\neq\left\|V_w-\frac{\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)}{\sum_{j=0,j\neq w}^{n}l_j}\right\| \quad (5)$$

Formula 5 is the necessary and sufficient conditions for the simplex not to degrade after each vertex deflects the same distance.

Definition: the maximum edge vector of the simplex $S=(s_1,\ s_2,\cdots,\ s_u,\cdots,\ s_n)^T$,

$s_u=\max_{i=0}^{n}\max_{j=i+1}^{n}(|v_{iu}-v_{ju}|)$. It reflects the maximum of all edge vectors; the minimum edge vector $T=(t_1,\ t_2,\cdots,\ t_u,\cdots,\ t_n)^T$,

$t_u=\min_{i=0}^{n}\min_{j=i+1}^{n}(|v_{iu}-v_{ju}|)$. It reflects the m minimum of all edge vectors; then

$$\left\|V_w-\frac{\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)}{\sum_{j=0,j\neq w}^{n}l_j}\right\|\geq\left|\|V_w\|-\frac{\left\|\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)\right\|}{\left|\sum_{j=0,j\neq w}^{n}l_j\right|}\right|\geq\left|\|V_w\|-\frac{\left\|\sum_{j=0,j\neq w}^{n}l_jS\right\|}{\left|\sum_{j=0,j\neq w}^{n}l_j\right|}\right|=\|V_w\|-\|S\|,$$

$$\left\|V_w-\frac{\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)}{\sum_{j=0,j\neq w}^{n}l_j}\right\|\leq\|V_w\|+\frac{\left\|\sum_{j=0,j\neq w}^{n}l_j(V_j-V_w)\right\|}{\left|\sum_{j=0,j\neq w}^{n}l_j\right|}\leq\|V_w\|+\frac{\left\|\sum_{j=0,j\neq w}^{n}l_jS\right\|}{\left|\sum_{j=0,j\neq w}^{n}l_j\right|}=\|V_w\|+\|S\|$$

, therefore when $r_j\leq\|\|V_w\|-\|S\|\|$ or $r_j\geq\|V_w\|+\|S\|$ (6)

the simplex is not degraded after the deflection operation. Formula (6) is the sufficient but not necessary condition for the simplex not to degrade after the deflection operation. Combined with this condition, the algorithm is relatively close to the extreme point in later times of iteration when searching and the amount of deflection should be reduced. Set rM as the amount of deflection at the beginning and rm as that of the last generation, then the amount of deflection can be determined with linear decreasing relationship:

$$r=\left(r_M+\frac{r_m-r_M}{T_{\max}}t\right)\|S\| \quad (7)$$

At the same time, set the number of the vertices slightly bigger than n+1 in order to prevent degradation.

### *B. Introduction of the symmetry operation*

Usually the optimal vertex Vb is closer to the target extremum than the center point, and the direction of each vertex Vj and Vb tend to the extremum faster than the direction along the central point [13]. So set the symmetric point of each vertex Vj about the optimal point Vb is Yj, that is:

$$Y_j=V_b+\delta_j(t)(V_b-V_j),j=0,1,\cdots,n\wedge j\neq b \quad (8)$$

Replace Vj with Yj if Yj is better than Vj, then multiple vertices can be replaced in this way to accelerate convergence. Symmetric length is computed according to formula（9）:

$$\delta_j(t) = \max\left( \mu\left( \frac{f_j}{\sum_{i=0, i\neq b}^{n} f_i} - \frac{1}{n} \right) + s(t), s_{\min} \right) \tag{9}$$

In formula (9), *S(t)* is a degressive non negative function of iterative generation number t, which not only avoids the possibility of a dead cycle about continuous symmetry at the same vertex but also produces the effect of speeding up the convergence which is the same as that of edge shrinkage. It is usually expressed with a linear relation as follows:

$$s(t) = s_{init} - kt \tag{10}$$

In the formula, $f_j$ is the adaptive value of vertex Vj; $s_{init}$, $s_{min}$, *K* is a constant bigger than 0. Constant μ is used to fine tune the effect of the ratio of vertex adaptive value. In summary, the simplex does not degrade if a simplex still constitutes a simplex after the symmetry operation.

*C. The adaptive modified reflection direction and reflection coefficient*

Sometimes the line direction between the worst point and the central point is very different from the descent direction, and this direction should be modified with the optimizing direction. When reflecting on the vertex Vi, first reflect as usual along the center Ci except Vi to R ', and then deflect distance D to the optimizing direction, taking Vw as an example to reflect the process. The reflection point Ri is expressed as follows:

$$R_i = C_i + \alpha(t)(C_i - V_i) + d(t)\vec{E}(t), C_i = \frac{1}{n}\sum_{j=0, j\neq i}^{n} V_j \tag{11}$$

Set m vertices are replaced with reflection points, the new vertices are V0，V1，…，Vn-m，Rn-m+1，R n-m+2，…，Rn. li is any n numbers which are not all 0, then discuss about the conditions the simplex does not degrade after modified reflection: Set the above formula is not equal to 0, then

$$d\vec{E}\sum_{i=n-m+1}^{n} l_i \neq -\sum_{i=1}^{n-m}\left( l_i + \frac{1+\alpha}{n}\sum_{j=n-m+1}^{n} l_j \right)(V_i - V_0) - \sum_{i=n-m+1}^{n}\left( \frac{1+\alpha}{n}\sum_{j=n-m+1, j\neq i}^{n} l_j - d\varepsilon_i \right)(V_i - V_0) \tag{12}$$

Formula (12) is the necessary and sufficient conditions the simplex does not degrade after the reflection operation. It is also the constraint condition of offset d. As Vi − Vw is n n-dimensional vector of linearly independent, the optimization direction $\vec{E}$ can be linearly represented as follows:

$$\vec{E} = \sum_{i=1}^{n} \varepsilon_i (V_i - V_0) \tag{13}$$

$$\sum_{i=1}^{n-m} l_i (V_i - V_0) + \sum_{i=n-m+1}^{n} l_i (R_i - V_0)$$
$$= \sum_{i=1}^{n-m}\left( l_i + \frac{1+\alpha}{n}\sum_{j=n-m+1}^{n} l_j + d\varepsilon_i \sum_{j=n-m+1}^{n} l_j \right)(V_i - V_0) + \sum_{i=n-m+1}^{n}\left( \frac{1+\alpha}{n}\sum_{j=n-m+1}^{n} l_j - \frac{1-m\alpha+\alpha}{n} l_i + d\varepsilon_i \sum_{j=n-m+1}^{n} l_j \right)(V_i - V_0)$$

Set the above formula is equal to 0, then

$$l_i + \left( \frac{1+\alpha}{n} + d\varepsilon_i \right)\sum_{j=n-m+1}^{n} l_j = 0,\ i = 1, 2, \cdots, n-m \tag{14}$$

$$\left( \frac{1+\alpha}{n} + d\varepsilon_i \right)\sum_{j=n-m+1}^{n} l_j - \frac{1-m\alpha+\alpha}{n} l_i = 0,\ i = n-m+1, n-m+2, \cdots, n \tag{15}$$

Add up m equations determined in formula (15) from i=n − m + 1 to n and get the following formula:

$$\left( \frac{(m-n-1)\alpha + m - 1}{n} + d\sum_{i=n-m+1}^{n} \varepsilon_i \right)\sum_{i=n-m+1}^{n} l_i = 0$$

If $\sum_{i=n-m+1}^{n} l_i = 0$, then it is known from formula (14) and formula (15) that li=0，i=1，2，…，n; Then the simplex does not degrade and

$$\frac{(m-n-1)\alpha + m - 1}{n} + d\sum_{i=n-m+1}^{n} \varepsilon_i \neq 0 \tag{16}$$

Formula (16) is a sufficient but not necessary condition the simplex does not degrade after modified reflection. Now the value range of modified offset D will be estimated. n points are all replaced in the extreme case, namely m=n, then formula (14) does not exist and formula (16) is simplified as follows:

$$d\sum_{i=1}^{n} \varepsilon_i \neq \frac{1-n+\alpha}{n} \tag{17}$$

Take the norm from formula (13), combine the maximum and minimum edge vector to zoom the inequality and get the following formula:

$$\frac{1}{\|S\|} \le \left| \sum_{i=1}^{n} \varepsilon_i \right| \le \frac{1}{\|T\|}$$

as d>0，combine with formula (17) and get formula (18):

$$d \le \left| \frac{1+\alpha}{n} - 1 \right| \|T\| \quad \text{or} \quad \left| \frac{1+\alpha}{n} - 1 \right| \|S\| \le d \tag{18}$$

Formula (18) is a sufficient but not necessary condition the simplex does not degrade after the reflective deflection, as long as d meets formula (18), the simplex will not degrade after the reflective deflection.

If the search speed is fixed, the step size appears too small in the early stage when it is far from the extreme point while the step size appears too big in the late stage when it is near the extreme point which leads to inaccurate positioning and even the increase of edge shrinkage operation. This consumes unnecessary computations, thus the reflection coefficient decreases linearly with the generation number and changes the step size adaptively to speed up convergence:

$$\alpha(t) = \alpha_{init} - \frac{\alpha_{init} - \alpha_{end}}{T_{max}} t$$

$\alpha_{init}$ and $\alpha_{end}$ are the starting and ending reflection coefficients respectively (19). In practical application, the offset D is taken as a certain function of reflection coefficient α(t). For example, it is in proportional toα(t):

$$d(t) = \theta \alpha(t) \|T\| \tag{19}$$

*D. Shrink the edge proportionally*

Shrinking according to the adaptive value ratio of each vertex can be close to the optimal point faster when shrinking the edge, the new vertex Yi after shrinkage edge is expressed as follows:

$$Y_i = V_i - \beta_i (V_i - V_b) \text{ or } Y_i = V_w + \beta_i (V_i - V_w) \tag{20}$$

The shrinkage edge ratio of each vertex βi is:

$$\beta_i = \frac{f_i}{\sum_{j=0, j\ne b}^{n} f_j} - \frac{1}{n} + \eta, \quad \eta \in (0,1) \tag{2}$$

In the formula,η is the shrinkage edge ratio corresponding to the vertex whose adaptive value is equal to the average. The ratio is better in the range of 0.25－0.35 and n is vertex number minus 1.

*E. Procedures of the algorithm*

The procedures of the improved simplex algorithm are as follows:

1) Initialize the parameters.

2) Generate the initial simplex. The vertex can be specified manually or generate randomly according to $V_i = V_0 + \lambda_i e_i$ when V0 is within the boundary.

3) Compare the adaptive values to determine the worst vertex Vw, the best vertex Vb, the relatively worst Vp.

4) Compute the central point, the maximum and minimum edge, and compute the optimization direction according to formula (1) and formula (2).

5 )Implement the deflection operation.

5.1Compute the amount of deflection r according to formula (6) and formula ( 7) ;

5.2 If i≠w, compute the deflection Yi for vertex Vi according to formula (3)

5.3 If Yi is better than Vi, replace Vi with Yi.

5.4 If the deflection has been operated on all vertices, then turn to 6; otherwise turn to 5.2.

6) Implement the symmetry operation.

6.1 Compute the adaptive value ratio for each vertex.

6.2 If i≠b, compute the symmetrical point Yi for vertex Vi according to formula (8), (9) and formula ( 10)

6.3If the Yi is better than Vi, receive Yi.

6.4 If the symmetry has been operated on all vertices, then turn to 7; otherwise turn to 6.1.

7) Implement the reflection operation.

7.1 Compute the reflection coefficient α(t) and anti-offset d(t) according to formula (18), (19) and formula (20).

7.2 Compute the reflection point R'w for vertex Vw according to the standard algorithm.

7.3 Then deflect R'w to Rw, according to (11), replace Rw with R'w if R'w is better than Rw

7.4 If Rw is better than Vp, replace Vw with Rw and turn to 10, otherwise turn to 8.

8) Implement the contraction operation.

8.1 If the reflection point Rw is worse than Vw, contraction point Vs= Cw－β(Cw－Vw)，otherwise Vs= Cw＋β(Cw－Vw).

8.2 If Vs is better than Vp, replace Vw with Vs and turn to 10 , otherwise turn to 9.

9) Implement the edge shrinking operation on each vertex.

9.1 Compute shrinkage edge ratio according to formula (22).

9.2 Implement the edge shrinking operation on vertex Vi according to formula (21)

9.3 Turn to 9.1 to perform a loop execution until all vertices have been executed on.

10) Test whether the operation meets the convergence

criteria. If $\left\{\frac{1}{n+1}\sum_{i=0}^{n}\left(f(V_i)-f(\overline{V})\right)\right\}^{\frac{1}{2}} \le \varepsilon$, then stop, otherwise turn to 3 for a loop.

## III. EXPERIMENTAL VERIFICATION

This paper takes the function's computational times and its iterative algebra as the main standard to measure the performance of the algorithm when it reaches the specified precision, because function solution is the main computation for such simplex direct optimization algorithm to compute complex or high-dimensional-function. The tested functions are shown in table 1:

TABLE 1. TESTED FUNCTIONS IN THIS PAPER

| | |
|---|---|
| Function1：Sphere | $f_1(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n}x_i^2; f_4^*(0,0,\cdots,0)=0; x_1,x_2,\cdots,x_n\in R$ |
| Function 2：Quadratic | $f_2(x_1,x_2,\cdots,x_n)=\sum_{i=1}^{n}i\times x_i^4; f_2^*(0,0,\cdots,0)=0; x_1,x_2,\cdots x_n\in R$ |

Add up the angle $\arccos\frac{\langle \vec{E}(t),-\nabla f(V_w(t))\rangle}{\|\vec{E}(t)\|\|\nabla f(V_w(t))\|}$ between each generation $\vec{E}(t)$ and the steepest descent direction $-\nabla f(V_w(t))$ as the deviation between the optimizing direction and the steepest descent direction to measure the descending degree of optimizing direction. Choose different forgetting rateρ,take function1 and function 2 as examples, when rM=0.3,rm=0.1, the initial step sizeλ=2,Sinit=1.5,k=-0.015,Smin=0.25,μ=1,αinit=1.4,αend =0.5,θ=0.3, the contraction coefficientβ=0.2,η=0.25, Tmax=128 and each weight of x0 is 12, change the functions' dimension number and use the functions which reach specified precision 1e-8 to compute the times and iterative algebra so as to test the effect ρand optimizing direction have on the effect of the algorithm. In the experiment ρincreases from 0.01 to 0.9 according to step size 0.02, average the first 5 results each function's cases compute the least, which is shown in table 2.

TABLE 2 THE EFFECT THE FORGETTING RATE P HAS ON THAT OF THE ALGORITHM

| dimension number / parameter | | 10 | 40 | 70 | 100 |
|---|---|---|---|---|---|
| funtion1 | forgetting rate ρ | 0.406 | 0.574 | .244 | .238 |
| | computational times | 1643.4 | 6539.6 | 11371.8 | 16015.8 |
| | iterative algebra | 68.2 | 77 | 77.4 | 76.8 |
| | deviation angle from the steepest descent direction | 3860.193 | 5060.538 | 2871.260 | 3032.714 |
| function 2 | forgetting rate ρ | 0.47 | 0.492 | 0.344 | 0.516 |
| | computational times | 1314.6 | 6002.6 | 10321.4 | 14947 |
| | iterative algebra | 54.4 | 71 | 70.8 | 72.8 |
| | deviation angle from the steepest descent direction | 3278.786 | 4061.903 | 3138.356 | 4379.231 |

It can be known from the above table: the more appropriate range of ρ is within 0.25-0.55 and better effects can be obtained at this time. It is also found in the experiment that: the smaller ρ is, the closer the optimizing direction is to the steepest descent direction. But at this time the computational times is not the least and the overall effect of the algorithm is not the best. Because the smaller ρ is, the more slowly the optimizing direction changes and the more

approximate it is to local properties and the gradient is just the local property of functions. When ρ is smaller, the deviation angle from the steepest descent direction is naturally smaller. When the optimizing direction tends to be localized, even fall quickly, the overall effect is not good because the searching step size is small and the searching path may appear twists and turns[8],this is also the reason why the gradient method converges slowly. In other words, the appropriate value ρ can obtain faster speed than that of the gradient method.

In order to verify the performance of the deflection operation and the range of deflection amount rM, change the range of rM for function 1, 2 under the condition in which the initial step size is 2 and 1.5, other parameters are the same as that in table 2 and use the functions which reach the precision of 1e-8 to compute the times and iterative algebra as a standard, the results are shown in table 3.

TABLE 3 DIFFERENT DEFLECTION EFFECTS CORRESPONDING TO DIFFERENT DEFLECTION AMOUNT

| test case | | | $r_M$=0.3 | | | $r_M$=1.3 | | | $r_M$=2.3 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Function | Starting point | Dimension number | Computational times | iterative algebra | Vertex number of deflection | Computational times | iterative algebra | Vertex number of deflection | Computational times | iterative algebra | Vertex number of deflection |
| Function1 | 12 | 10 | 1712 | 71 | 128 | 1912 | 77 | 120 | 2057 | 78 | 164 |
| | | 40 | 6846 | 77 | 535 | 7234 | 81 | 601 | 7445 | 85 | 352 |
| | | 70 | 11650 | 78 | 757 | 13074 | 86 | 945 | 13437 | 88 | 801 |
| | | 100 | 16517 | 79 | 1274 | 204408 | 96 | 902 | 21625 | 100 | 892 |
| Function2 | 1000 | 10 | 1500 | 62 | 88 | 1580 | 65 | 92 | 1841 | 74 | 99 |
| | | 40 | 6253 | 74 | 415 | 6372 | 73 | 600 | 7667 | 87 | 320 |
| | | 70 | 11303 | 78 | 665 | 12932 | 87 | 629 | 12788 | 83 | 581 |
| | | 100 | 15093 | 73 | 545 | 35185 | 171 | 800 | 17527 | 82 | 875 |

It can be known from table 3: The computational times of functions and the vertex number of deflection are obviously better when rM<1.Ideal effects can usually be obtained when rM is near 0.2 or 0.3; When rM>1, the possibility of simplex degradation increases and the effect is relatively poor, and the effect of the algorithm and the value of rM are uncertain. It can also be seen that the bigger the dimension number is, the better the deflection effect. The deflection effect is not obvious when the dimension number is small. In addition, although the deflection effect is usually very good when Vw is taken as the starting point of deflection, sometimes better deflection effect can be obtained when Vb is taken as the starting point of deflection.

As for the dimension number, the bigger the dimension number is, the more obvious the effects of the symmetry operation are and the better the cost performance of the algorithm is. The results can be obtained in no more than 20 generations in a 100-dimensional-function, which is what the standard SM cannot imagine. In the experiment, it is very difficult for SM to find a solution when the function is more than 10 dimensional and it is almost impossible for SM to find a solution when the function is beyond 20 dimensional. It is particularly noteworthy that when Sinit is appropriate, the times of reflection is very small in the process of computation. The whole process of finding a solution is through symmetry operations almost without other operations, namely, a solution can be found only through symmetry operations. The experiments showed that in using SM for high-dimensional-functions there appears a phenomenon that the reflection point is better than the best point but it is very difficult to be more optimal than the best point. Thus the reflection point still remains near the primal simplex, the evolution is slow and the reflection results are poor. The symmetry operation can be viewed as a reflection in n different directions. It increases the searching direction and can find a better descent direction, which compensates the problem of inefficient reflection effect for high-dimensional-functions. This is particularly effective for high-dimensional-functions, and the higher-dimensional-function test is shown in table 5.

TABLE 4 EFFECTS OF THE ALGORITHM FOR DIFFERENT SYMMETRY LENGTH

| Function | $S_{init}$ | Measurement criteria | 10 Dimensions | 40 Dimensions | 70 Dimensions | 100 Dimensions |
|---|---|---|---|---|---|---|
| Function 1 | 0.3 | Computational times | 413 | 1441 | 3011 | 3501 |
| | | Iterative algebra | 17 | 15 | 19 | 15 |
| | | Vertex number of symmetry | 142 | 430 | 1067 | 896 |
| | | Deviation from the steepest descent direction | 28.409 | 24.338 | 32.726 | 22.545 |
| | | Times of reflection | 0 | 1 | 1 | 1 |
| | 1.2 | Computational times | 1458 | 5985 | 11223 | 14997 |
| | | Iterative algebra | 63 | 73 | 78 | 72 |
| | | Vertex number of symmetry | 224 | 1149 | 1648 | 3040 |
| | | Deviation from the steepest descent direction | 819.365 | 866.016 | 862.866 | 889.326 |
| | | Times of reflection | 3 | 17 | 49 | 5 |
| | 2.5 | Computational times | 1982 | 7805 | 13521 | 19339 |
| | | Iterative algebra | 86 | 95 | 94 | 94 |
| | | Vertex number of symmetry | 140 | 720 | 1536 | 2249 |
| | | Deviation from the steepest descent direction | 2834.428 | 2642.035 | 2556.184 | 2672.150 |
| | | Times of reflection | 18 | 37 | 61 | 63 |
| Function 2 | 0.3 | Computational times | 288 | 1042 | 1962 | 2800 |
| | | Iterative algebra | 12 | 12 | 13 | 13 |
| | | Vertex number of symmetry | 99 | 410 | 766 | 1205 |
| | | Deviation from the steepest descent direction | 18.823 | 19.806 | 17.307 | 19.284 |
| | | Times of reflection | 0 | 0 | 0 | 2 |
| | 1.2 | Computational times | 1202 | 5394 | 9589 | 14096 |
| | | Iterative algebra | 52 | 67 | 68 | 70 |
| | | Vertex number of symmetry | 148 | 872 | 1479 | 2093 |
| | | Deviation from the steepest descent direction | 590.520 | 689.035 | 676.203 | 684.261 |
| | | Times of reflection | 6 | 8 | 3 | 6 |
| | 2.5 | Computational times | 1656 | 7190 | 13070 | 18078 |
| | | Iterative algebra | 72 | 89 | 92 | 89 |
| | | Vertex number of symmetry | 61 | 712 | 810 | 1883 |
| | | Deviation from the steepest descent direction | 2232.580 | 2443.050 | 2348.715 | 2334.978 |
| | | Times of reflection | 12 | 12 | 22 | 24 |

TABLE 5 TEST RESULTS FOR HIGH-DIMENSIONAL-FUNCTIONS

| Test case | | parameter | | | | | | | Test results | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Function | Dimension number | ρ | $r_M$ | $r_m$ | $S_{init}$ | $S_{min}$ | k | $\alpha_{init}$ | Computational times | Iterative algebra | Achieved precision | Minimum |
| Function 1 | 150 | 0.5 | 0.3 | 0.1 | 0.1 | 0.1 | 0.01 | 1.2 | 3302 | 9 | 5.989E-09 | 1.417E-06 |
| | 200 | 0.2 | 0.2 | 0.1 | 0.1 | 0.1 | 0.01 | 0.8 | 4399 | 9 | 5.585E-09 | 1.935E-07 |
| | 250 | 0.3 | 0.1 | 0.1 | 0.3 | 0.25 | 0.03 | 0.8 | 7501 | 13 | 3.033E-09 | 4.6E-09 |
| | 300 | 0.1 | 0.4 | 0.1 | 0.1 | 0.1 | 0.01 | 1.2 | 6901 | 8 | 4.913E-09 | 2.74E-08 |
| Function 2 | 150 | 0.2 | 0.1 | 0.1 | 0.1 | 0.1 | 0.01 | 1 | 2702 | 8 | 5.111E-09 | 2E-10 |
| | 200 | 0.3 | 0.1 | 0.1 | 0.2 | 0.2 | 0.01 | 0.9 | 3802 | 9 | 7.033E-09 | 8.21E-08 |
| | 250 | 0.5 | 0.1 | 0.1 | 0.3 | 0.25 | 0.03 | 1.1 | 5001 | 9 | 4.688E-09 | 2.9E-09 |
| | 300 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.01 | 1.2 | 5402 | 8 | 1.088E-09 | 5E-10 |

Hybrid particle swarm algorithm proposed in literature [15] executes Hooke-Jeeves local search and the mutation strategy according to the probability μd for 5 dimensional function 1 and 2 dimensional function 2 in the initial range of $\pm 20$ and $\pm 50$ respectively, the execution results are shown in table 6. And for the same function, the optimal solution results of SM and ASM in the same test mode mentioned above are shown in table 7, where $\alpha$、$\beta$、$\gamma$ are the coefficients of SM's reflection, compression and expansion. It can be seen from table 7 that the computational times and iteration algebra ASM requires are smaller than that of SM, which shows that the adaptive simplex method consumes less computation amount and less time for low- dimensional-function than the standard simplex algorithm , thus it is more efficient. The table also shows that the reflection coefficient of the standard algorithm is smaller, especially when it is less than 1 and when $\alpha + \gamma \in [1.5,2.5]$ , the effects are the best. That is, the smaller reflection coefficient and the bigger expansion coefficient are better combination. Comparing table 6 with table 7, it can be seen that there exists great disparity between the efficiency of hybrid particle swarm algorithm and that of ASM. For unimodal function optimization problem, the traditional algorithm is more effective than the

swarm intelligence algorithm and it is difficult to show the advantages of swarm intelligence algorithm.

TABLE 6 COMPARISON BETWEEN THE OPTIMAL RESULTS OF THE STANDARD ALGORITHM AND THOSE OF THE ADAPTIVE ALGORITHM

| Test case | | | Standard simplex optimization method | | | | | Adaptive simplex optimization method | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Function | Dimension number | Starting point | α | β | γ | Computational times | Iterative algebra | ρ | $r_M$ | $r_m$ | $S_{init}$ | $S_{min}$ | $\alpha_{init}$ | Computational times | Iterative algebra |
| Function 1 | 5 | 20 | 0.3 | 0.6 | 1.3 | 164 | 77 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.8 | 157 | 10 |
| Function 2 | 2 | 50 | 0.1 | 0.5 | 1.3 | 65 | 26 | 0.2 | 0.1 | 0.1 | 0.2 | 0.2 | 1.3 | 59 | 7 |

## IV. CONCLUSION

The adaptive simplex method (ASM) is proposed in this paper based on the research on the search mechanism of the simplex method (SM). This algorithm guides the current search direction according to the most recent search history to reduce the blindness in finding a solution. The experimental results show that ASM can not only reduce computational times of functions and converge faster than SM but also overcome the disadvantage of SM's poor ability for high- dimensional-functions. Its advantages in solving the problem of high-dimensional-functions are more obvious.

## REFERENCE


[1] [1] Avriel M . Nonlinear programming : analysis and methods[M]. Prentice-Hall, Englewood Cliffs, NJ, 1976

[2] [2] W . Spendly , G . R . Hext , F . R . Himsworth . Sequential application of simplex designs in optimization and evolutionary operation[J]. Technometrics, 1962（4）: 441－461

[3] [3] J A Nelder, R. Mead. A simplex method for function minimization[J]. Computer J. 1965, （7）: 308－313

[4] [4] M. J. Box. A new method for function minimization[J]. Comput.J. 1965 : 42－52

[5] [5] Jeffrey C. Lagarias, James, A. Reeds, Margaret H. Wright, et al. Convergence Properties of the Nelder－Mead Simplex Method in Low Dimensions[J] . SIAM Journal of Optimization, 1988 : 112-147

[6] [6] Gerdts M . Direct Shooting Method for the Numerical Solution of Higher-Index DAE Optimal Control Problems[J] . Journal of Optimization Theory and Applications 2013, 117（2）: 267－294

[7] [7] CHEN Guo-chu, YU Jin-shou. Simplex Particle Swarm Optimization Algorithm and Its Application [J]. Journal of System Simulation, 2006, 18（4）: 862－865

[8] [8] [Cheng shuihui,LI hahong. The complex method to solve multidimensional nonlinear constrained optimization problem [J] . Precise Manafacturing & Automation, 2012, 3 : 37－38

[9] [9] LI Liang, CHI Shi-chun, LIN Gao . Modified complex method and its application to the slope stability analysis [J]. Journal of Harbin Institute of Technology, 2015, 37（10）: 1429－1432

[10] [10] Luo wenguang . Research on An Improved Simplex Algorithm and its Simulation [J]. Computer Simulation, 2013, 20（10）: 62－64

[11] [11] HAN Zhi-xin, DU Jin-xia, LIU Bin, FAN Xin-qiang, WANG Shao-feng . A Revolving Direction Search Algorithm of the Complex Shape on the Nonlinear Optimization Problems [J]. Journal of Gansu Sciences, 2015, 19（3）: 103－106.

[12] [12] Zhang xiaohong ,Zhang junfu, Zengyuyu, An Improved Algorithm of Complex Method in Optimization [J] Journal of Sichuan University of Science and Technology