Computer Science and Application 计算机科学与应用, 2021, 11(11), 2682-2699
Published Online November 2021 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2021.1111272

Hans汉斯

基于协方差矩阵特征值和特征向量的差分进化算法改进研究

石宏庆[1]，侯　庆[1,2]

[1]贵州省通信产业服务有限公司，贵州　贵阳
[2]贵州大学计算机科学与技术学院，贵州　贵阳

收稿日期：2021年10月5日；录用日期：2021年11月4日；发布日期：2021年11月12日

摘　要

差分进化算法是一种具有性能强、简单易用和自适应能力强等优点的全局优化算法，但同时也存在早熟收敛问题和搜索停滞问题，因此本文的研究目的在于研究协方差矩阵的特征值和特征向量对差分进化算法搜索性能的影响，探究特征值和特征向量以何种改进策略能够提高算法搜索性能，这对算法适应更为复杂的优化问题和满足更高的求解质量有非常重要的研究意义。本文使用协方差矩阵的特征值和特征向量改进差分进化算法中初始种群和变异个体的计算规则，通过仿真实验证明，特征值按照本文的改进策略未能提高差分进化算法的搜索性能，而特征向量在一定的种群进化代数内能够正确的引导差分进化算法进行搜索，并有效提高算法搜索性能。

关键词

差分进化算法，协方差矩阵，特征值，特征向量

Research on Improvement of Differential Evolution Algorithm Based on Eigenvalues and Eigenvectors of Covariance Matrix

Hongqing Shi[1], Qing Hou[1,2]

[1]Guizhou Communication Industry Service Co., Ltd., Guiyang Guizhou
[2]College of Computer Science and Technology, Guizhou University, Guiyang Guizhou

Received: Oct. 5th, 2021; accepted: Nov. 4th, 2021; published: Nov. 12th, 2021

Abstract

The differential evolution algorithm is a global optimization algorithm with strong performance,

文章引用: 石宏庆, 侯庆. 基于协方差矩阵特征值和特征向量的差分进化算法改进研究[J]. 计算机科学与应用, 2021, 11(11): 2682-2699. DOI: 10.12677/csa.2021.1111272

easy to use and strong adaptability. However, there are also premature convergence problems and search stagnation problems. Therefore, the research of this paper aims to study the influence of the eigenvalues and eigenvectors of the covariance matrix on the search performance of the differential evolution algorithm, and to explore how the improved strategies of eigenvalues and eigenvectors can improve the search performance of the algorithm, which is of great significance for the algorithm to adapt to more complex optimization problems and satisfy higher quality of solution. In this paper, the eigenvalues and eigenvectors of the covariance matrix are used to improve the calculation rules of the initial population and the mutated individual in the differential evolution algorithm. The experimental results show that eigenvalues according to the improved strategy of this paper cannot improve the search performance of the algorithm, and the eigenvectors can correctly guide the differential evolution algorithm in a certain population evolution algebra, and effectively improve the search performance of the algorithm.

Keywords

Differential Evolution Algorithm, Covariance Matrix, Eigenvalues, Eigenvector

1. 引言

在生活和工作中，人们对同一问题往往会有不同的解决方法，并且会通过多方论证从而得出最佳的解决方案，例如什么形状的储水罐才能达到用料最省呢？什么季节播种怎样的农作物才能使收成最大呢？等等诸如此类的问题我们通常称之为优化问题；除此外，优化问题也广泛存在于生产和科研的各个领域，如机械制造、生产控制、管理科学等等，当人们在生活与生产中遇到这样的优化问题，优化算法便由此孕育而生。

差分进化算法(Differential Evolution, DE)是一种基于种群遗传进化的全局优化算法[1]，其主要求解的问题是实数优化问题，其首次出现的时间是 1995 年，由 Storn 和 Price 提出[2]，属于演化算法的一种。差分进化算法是一种模拟生物进化的概率模型，在反复进化中，创造和存储适应环境的优质个体。差分进化算法的基本思想：算法首先在父代种群中随机选择三个个体以执行变异操作之后，变异个体与目标个体做交叉操作以生成新的子代个体，最终在父代和子代个体中做选择操作，选择满足要求的对象保存至下一代群体中。由此可看出，差分进化算法的工作步骤基本上与其他进化算法一致，主要包括变异(Mutation)、交叉(Crossover)和选择(Selection)三种操作；其中变异操作是父代种群中两个对象的随机选择，将其二者做向量差生成差分矢量加权，并且按照特定的规则和第三个对象求和而产生变异个体；交叉操作是通过将变异对象的参数与所选目标对象混合来创建试验个体；选择操作是比较试验个体和目标个体的适应度值，取其优者。

差分进化算法因具有结构简单、容易实现、收敛快速、搜索性能强等特点而被广泛应用在机械设计、信号处理、人工神经元网络、生物信息、电磁学等各个领域。虽然差分进化算法在很多领域上取得不错的成效，但是标准的差分进化算法在求解过程中随着进化代数的增加，会使种群的多样性变小，从而面临早熟收敛和搜索停滞等问题。同时随着科技飞速地发展，优化问题会越发的复杂，计算量将会越来越大，再使用标准的差分进化算法来求解优化问题会略显吃力。

因此针对该算法的不足之处，提出一个协方差矩阵与差分进化算法相结合的想法，把差分进化算法和基于特征向量的交叉算子结合，探索协方差矩阵的特征值和特征向量对算法搜索性能的影响，这对改进差分进化算法以适应更为复杂的优化问题和满足更高的求解质量有着非常重要的研究意义。

2. 国内外研究现状

差分进化算法具有非常很不错的全局寻优能力，再加上全球也掀起人工智能的浪潮，进化计算作为人工智能中重要组成模块，因而使得差分进化算法成为进化计算领域的研究热点之一。差分进化算法在工程、数学、计算机科学和物理这些基础学科研究输出了很多优秀的论文，这些论文占论文总数将近 70%，而其他所有学科只占了总数的 30%左右。由此可见，差分进化算法研究的领域主要集中在工程和数学应用领域[3]。

差分进化算法搜索性能改进研究主要针对差分进化算法的两大缺陷来进行：一是搜索停滞问题，即算法在种群进化过程中，无法继续寻找全局最优解，停止向全局最优解的方向进行进化的现象；二是早熟收敛问题，即随着种群的进化代数增加，种群多样性减少，导致过早局部最优解的现象。为解决算法中出现上述的现象，针对差分进化算法的改进主要围绕以下几方面来进行：控制参数设置[4]、变异策略选择[5]、种群结构和混合优化算法。

通过阅读文献发现，差分进化算法已拓展至参数动态调整的多目标优化领域。文献[6]提出了 AMODE 算法，是一种基于变异性和交叉率动态调整的改进算法，该改进算法通过进化过程信息设计变异率和交叉率的动态调整策略，实现算法参数的自适应调整，以致达到了提高算法的全局寻优能力。在 MDADE 算法中，通过动态调整参数值使得算法能够适应种群的不同进化时期，同样达到了提高算法收敛速度的效果[7]。在 DMSDE 算法中，改进其交叉算子，有效地解决了算法容易陷入局部最优解范围的问题，提高了算法的收敛速度[8]。

3. 相关工作

3.1. 标准差分进化算法

标准差分进化算法具有结构简单，容易实现，搜索性能强等优点[9]，标准差分进化算法的操作流程如图 1 所示：

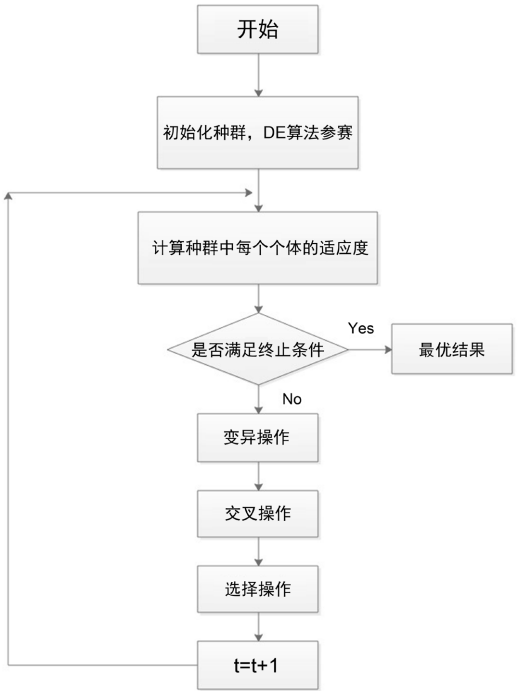


Figure 1. Calculation flow chart of standard differential evolution algorithm
图 1. 标准差分进化算法计算流程图

标准差分进化算法主要操作步骤如下[10]：

3.1.1. 初始化种群

根据给出的优化问题及条件，建立数学模型，给出差分进化算法和种群初始化的一些参数控制，标准差分进化算法初始化种群的产生方式如式(1)所示：

$$x_{j,i}(0)=x_{j,i}^{L}+rand(0,1)\times\left(x_{j,i}^{U}-x_{j,i}^{L}\right) \tag{1}$$

式(1)中：x_j^L 和 x_j^U 分别代表第 j 个个体 x_j 取值范围即 $\left[x_j^L,x_j^U\right]$；D 是解空间的维数，NP 代表的是种群的大小，$rand(0,1)$ 表示在(0, 1)范围内均匀分布的一个随机数[11]；NP 代表算法的种群信息量的大小，NP 值越大则意味着种群信息越丰富，导致计算量的增加无助于解决问题；相反，如果 NP 值越小，则会影响种群多样性，这种限制对算法的全局寻优同样没有帮助，严重会使算法收敛至局部最优解，导致搜索停滞。

3.1.2. 变异操作

在标准差分进化算法中还有一个重要的参数：交叉概率 CR [12]，交叉概率的取值为常数；CR 在交叉操作中扮演着重要的角色，它反映了子代和父代之间的信息交换程度和中间变异体间的信息交换程度的大小，如式(2)所示：

$$u_{j,i}(g+1)=\begin{cases}v_{j,i}(g+1), & \text{if } rand(0,1)\le \text{CR or } j=j_{rand}\\ x_{j,i}(g), & \text{otherwise}\end{cases} \tag{2}$$

其中，j_{rand} 为 $[1,2,3,\cdots,D]$ 的随机整数；CR 取值越大，交换信息量的程度就越大；相反，CR 取值越低，交换信息量的程度也就越低，将会导致种群的多样性迅速减小，不利于算法的全局寻优。

3.1.3. 选择操作

在标准差分进化算法中，通过计算父代个体和交叉产生的新个体的适应度，然后比较其大小，选择适应度较小的个体作为优质个体，进入下一代种群[12]，其选择方式如式(3)所示：

$$x_i(g+1)=\begin{cases}u_i(g+1), & \text{if } f(u_i(g+1))\le f(x_i(g))\\ x_i(g), & \text{otherwise}\end{cases} \tag{3}$$

3.2. 协方差矩阵

首先理解一下协方差的概念，在概率论和统计学中，协方差(Covariance)是用来刻画两个随机变量的总体误差，反映的是变量之间的二阶统计特性，当协方差是正数时，两个随机变量互为正相关关系，反之，二者则互为负相关关系；当两个变量是相同的情况下，协方差即为方差，因此方差是协方差的一种特殊情况。计算两个随机变量 X_i 、 X_j 的协方差公式如式(4)所示：

$$cov(X_i,X_j)=E\left[\left(X_i-E(X_i)\right)\left(X_j-E(X_j)\right)\right] \tag{4}$$

而协方差矩阵在统计学与概率论中的定义为，协方差矩阵[13]的每个元素是各个向量元素之间的协方差。n 维随机变量 $X=(X_1,X_2,X_3,\cdots,X_n)^{\mathrm{T}}$ 的协方差矩阵定义如式(5)所示：

$$C(X)=(c_{i,j})_{n\times n}=\begin{bmatrix}cov(X_1,X_1) & \cdots & cov(X_1,X_n)\\ \vdots & \ddots & \vdots\\ cov(X_n,X_1) & \cdots & cov(X_n,X_n)\end{bmatrix}_{n\times n} \tag{5}$$

其中，$c_{i,j}=cov(X_i,X_j)$，显而易见，矩阵 C 是一个对称矩阵；协方差矩阵 C 中的对角线元素表示方差，非对角线元素表示随机向量 X 的不同分量之间的协方差。

3.3. 特征值和特征向量

不妨设 A 是 n 阶矩阵，如果数 λ 和 n 维非零向量 x 使关系式(6)

$$Ax = \lambda x \tag{6}$$

成立，则称数 λ 为矩阵 A 的特征值(Eigenvalues)，非零向量 x 称为 A 的对应于特征值 λ 的特征向量(Eigenvector)。式(6)可变换成式(7)：

$$(A - \lambda E)x = 0 \tag{7}$$

可通过计算式(7)，从而求解矩阵 A 的特征值和特征向量。

3.4. 向量组的线性组合

通常情况下，将若干个同维度的列向量(行向量)所组成的集合定义为向量组。设有向量组 $\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_m$，有一组实数 $k_1,k_2,k_3,\cdots,k_m$，使式(8)成立：

$$\beta = k_1\alpha_1 + k_2\alpha_2 + \cdots + k_m\alpha_m \tag{8}$$

则把向量 β 称为是向量组 $\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_m$ 的一个线性组合，或者称 β 可由向量组 $\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_m$ 线性表示。

而凸组合是一类特殊的线性组合，同时也是一种非负线性组合；当 $k_i \geq 0$，并且 $\sum_{i=1}^{n} k_i = 1$，则把 $\sum_{i=1}^{n} k_i\alpha_i$ 称为向量组 $\alpha_1,\alpha_2,\alpha_3,\cdots,\alpha_m$ 的一个凸组合(凸线性组合) [14]。

4. 仿真实验设计

4.1. 特征值和特征向量引导的差分进化算法

差分进化算法与遗传算法十分相似，都包括变异、交叉和选择操作，但是差分进化算法又有别于遗传算法，二者的主要区别在于变异操作上：在遗传算法中的变异操作是通过两个父代个体的交叉而产生新个体；在差分进化算法中的变异操作则是通过用两个个体的差分矢量与第三个个体求和得出新个体；因此变异操作是差分进化算法的关键一步。变异操作在构建变异个体的过程中是基于随机性，即参与变异操作的个体均为种群中的随机个体，如式(9)所示：

$$\mathrm{v}(:,\mathrm{m}) = \mathrm{x}(:,\mathrm{r1}) + \mathrm{F} * (\mathrm{x}(:,\mathrm{r2}) - \mathrm{x}(:,\mathrm{r3})) \tag{9}$$

式(9)中 r1、r2、r3 和 m 均为互不相等的正整数；从引导算法搜索的层面来看，此随机性对算法的引导性并不是很明确，可能会让算法在寻优过程中多走弯路，所以为算法探寻一个较为准确的搜索方向亦可提高算法的搜索性能。

在本文中，提出的特征值和特征向量源于差分算法解群体的协方差矩阵，解群体的协方差矩阵的特征值和特征向量携带当前搜索种群的所有信息，用其来改变差分算法的初始种群和变异个体的计算规则，分别为式(10)和式(11)：

$$\mathrm{x} = \mathrm{rand}(\mathrm{D},\mathrm{NP}) * \mathrm{b}(\mathrm{randi}([1,\mathrm{k}],1,1),1) + \mathrm{Xx} \tag{10}$$

$$\mathrm{v}(:,\mathrm{m}) = \mathrm{x}(:,\mathrm{r1}) + \mathrm{F} * (\mathrm{U}(:,\mathrm{randi}([1,\mathrm{D}],1,1))) \tag{11}$$

从改变后的计算规则可以看出，计算规则中的 randi()函数使得计算规则仍存在一定的随机性；从理论角度分析可得知，解群体的协方差矩阵的特征值和特征向量可能会给差分算法带来较为准确的搜索方向。

4.2. 标准差分进化算法计算测试函数

使用标准差分进化算法分别独立计算 Ackley 函数、Griewank 函数、Rastrigin 函数、Schaffer 函数和 Sphere 函数，其中算法参数种群数量 NP = 50，变量的维度 D = 20，最大进化代数 G = 1000，缩放因子 F = 0.6，交叉概率 CR = 0.9。

算法中有两个关键的取值点再标准差分进化算法中按照如式(12)和式(13)的计算规则取值：

$$x = \mathrm{rand}(D, NP) * (Xs - Xx) + Xx \tag{12}$$

式(12)中，x 为算法中定义的初始种群。

$$v(:, m) = x(:, r1) + F * (x(:, r2) - x(:, r3)) \tag{13}$$

式(13)中，$v(:, m)$ 为变异操作中产生的变异个体，将会被代入交叉操作产生试验个体，其中 $r1 \neq r2 \neq r3 \neq m$ 。

算法每次计算循环运行 20 次，得出测试函数计算耗时如表 1 所示：

Table 1. Time consuming for standard differential evolution algorithm to calculate test function (unit: s)
表 1. 标准差分进化算法计算测试函数耗时(单位：s)

运行次数	1	2	3	4	5	6	7	8	9	10
Ackley	1.14	0.81	0.46	0.49	0.48	0.45	0.43	0.45	0.44	0.46
Griewank	1.00	0.50	0.46	0.50	0.48	0.47	0.47	0.44	0.48	0.45
Rastrigin	1.01	0.48	0.47	0.46	0.51	0.46	0.46	0.46	0.47	0.45
Schaffer	1.00	0.79	0.42	0.40	0.42	0.41	0.41	0.40	0.40	0.40
Sphere	1.18	0.44	0.46	0.48	0.45	0.54	0.42	0.46	0.47	0.42
运行次数	11	12	13	14	15	16	17	18	19	20
Ackley	0.45	0.45	0.45	0.44	0.50	0.44	0.48	0.44	0.45	0.44
Griewank	0.47	0.45	0.47	0.45	0.48	0.45	0.47	0.46	0.47	0.46
Rastr	0.45	0.46	0.45	0.45	0.45	0.46	0.44	0.46	0.45	0.46
Schaffer	0.40	0.41	0.40	0.42	0.40	0.41	0.41	0.40	0.40	0.41
Sphere	0.43	0.43	0.43	0.43	0.42	0.43	0.43	0.43	0.43	0.42

表 1 中记录的时间为算法每循环一次的所耗时间。通过观察实验结果得知，标准差分进化算法程序第一次运行耗时明显偏高，其原因为：算法第一次运行，算法中变量需要赋值及计算，而随后的算法循环计算是在不清除前一次变量以及其值的情况下进行的，所以算法第一次运行耗时会高于其他次，但这并不影响后续算法改进研究。保存标准差分进化算法计算测试函数耗时，是为了与后续算法改进后的计算耗时作对比。

4.3. 整体观察特征值和特征向量对差分进化算法的影响

在其他实验环境不变的情况下，引入协方差矩阵，从而得出特征值和特征向量，用于改进标准差分进化算法中的初始种群和变异个体的计算规则。改进后差分进化算法的初始值和变异个体的计算规则如式(14)所示：

$$x = \mathrm{rand}(D, NP) * b(\mathrm{randi}([1, k], 1, 1), 1) + Xx \tag{14}$$

式(14)中，b 是特征值收纳的一个矩阵，从 b 中随机选取一个特征值代替掉变量上下限差值$(Xs-Xx)$，如式(15)所示：

$$v(:,m)=x(:,r1)+F*\left(U(:,randi([1,D],1,1))\right) \tag{15}$$

式(15)中，U 为特征向量组成的矩阵，从 U 中随机选取一个特征向量代替种群中两个随机个体产生的矢量差 $x(:,r2)-x(:,r3)$。

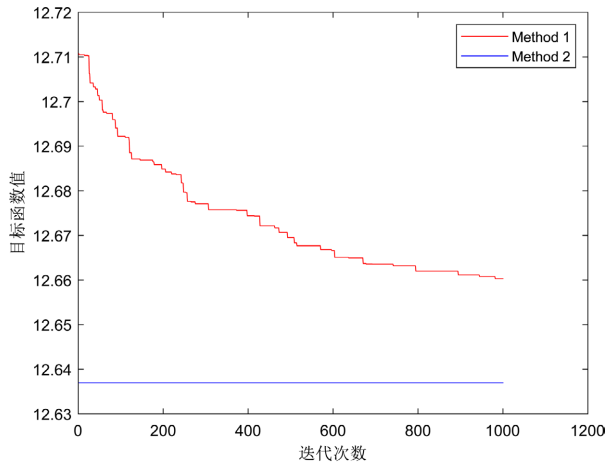
而在协方差矩阵的选取有两种方式：

Method 1：在 0~1 范围内随机生成 NP 行 D 列的矩阵，然后计算此矩阵的协方差矩阵，最后代入程序中计算其特征值和特征向量，此方法在一定程度上能够保持种群的多样。

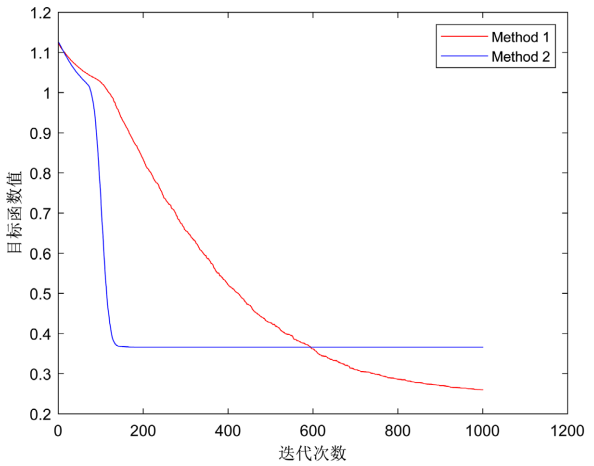
Method 2：使用标准差分进化算法计算目标函数时得出的解群体组成的矩阵，计算其协方差矩阵，然后代入程序进行计算。

分别将两种方式代入算法程序进行计算测试函数，算法每次计算循环运行 20 次，取其平均值，将其计算结果作比较分析，如图 2 所示。

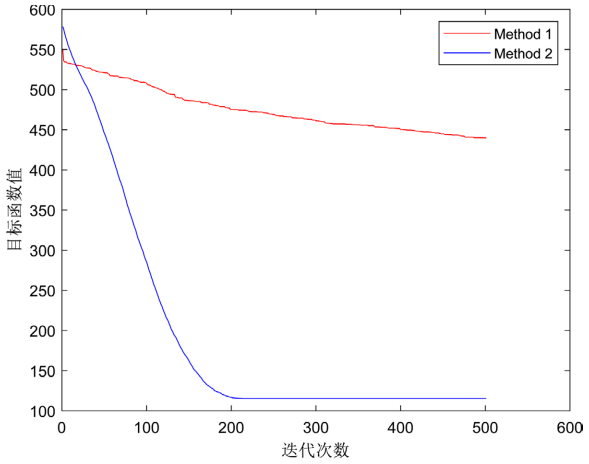
在图 2 中，红色曲线和蓝色曲线分别表示 Method 1 和 Method 2 对应的计算结果。观察实验结果，在 Ackley 和 Rastrigin 函数中，Method 2 的收敛速度比 Method 1 的收敛速度快，并且在同样的进化次数内，Method 2 能够收敛到比 Method 1 收敛到的局部最优解更好。Schaffer 函数则相反，Method 1 的收敛



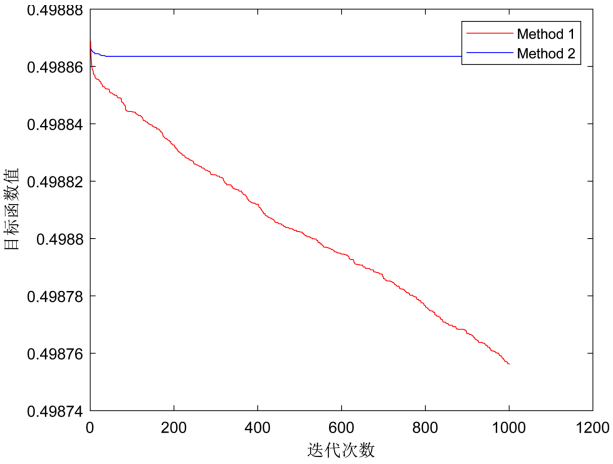
(a) Ackley 函数



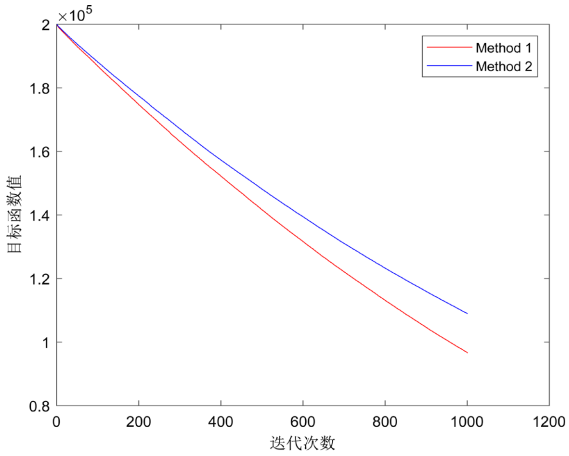
(b) Griewank 函数



(c) Rastrigin 函数



(d) Schaffer 函数



(e) Sphere 函数

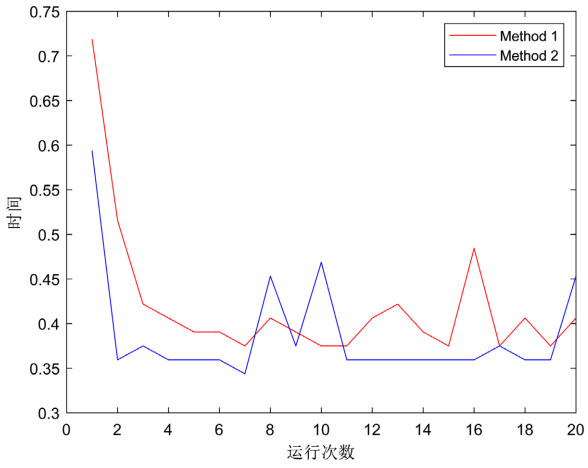
Figure 2. Convergence curves calculated in two methods
图 2. 两种方式计算的收敛曲线

速度相比于 Method 2 更快，收敛趋势也比 Method 2 的收敛趋势更乐观。在 Griewank 函数中，进化次数在 200 次前，Method 2 具有更好的收敛能力，但其容易陷入局部最优解，在进化次数为 600 次后，收敛效果不如 Method 1。Sphere 函数的两条收敛曲线呈现出相似的曲线形状，但 Method 1 的收敛速度优于 Method 2。

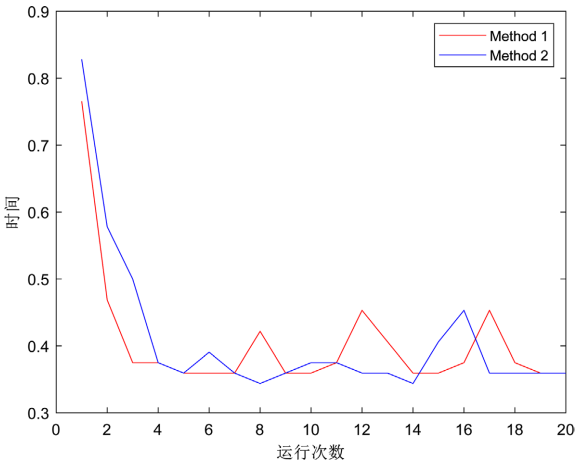
再把两种方式对算法运算耗时曲线作比较，分析两种方式对算法运算耗时的结果，如图 3 所示：

观察图 3 中的实验结果，发现两条曲线有多次相交，且波动幅度较小，说明两种方式对算法运算耗时相近。

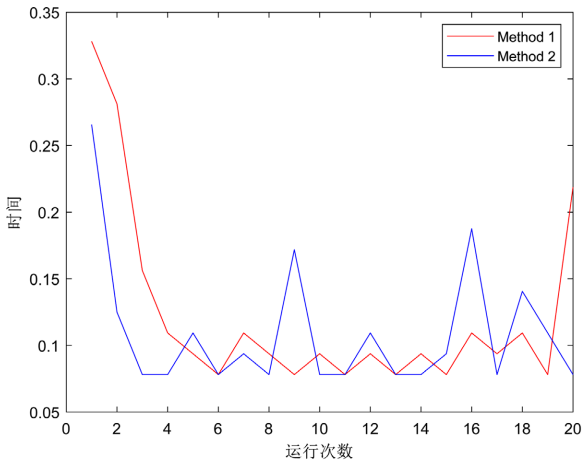
通过实验比较矩阵取值的两种方式，Griewank、Schaffer 和 Sphere 函数使用 Method 1 更适合算法寻优，因为使用 Method 1 的收敛速度以及曲线收敛趋势比 Method 2 更佳，同时两种方式对算法运算耗时相差较小，因此在后续仿真实验中，计算 Griewank、Schaffer 和 Sphere 函数时采用 Method 1，即在 0~1 范围内随机生成 NP 行 D 列的矩阵用于生成协方差矩阵。而 Ackley 和 Rastrigin 函数则与其他三个优化测试函数不同，使用 Method 2 更有利于差分进化算法计算 Ackley 和 Rastrigin 函数，因此在后续仿真实验中，计算 Ackley 和 Rastrigin 函数时采用 Method 2，即解群体组成的矩阵用来计算协方差矩阵。



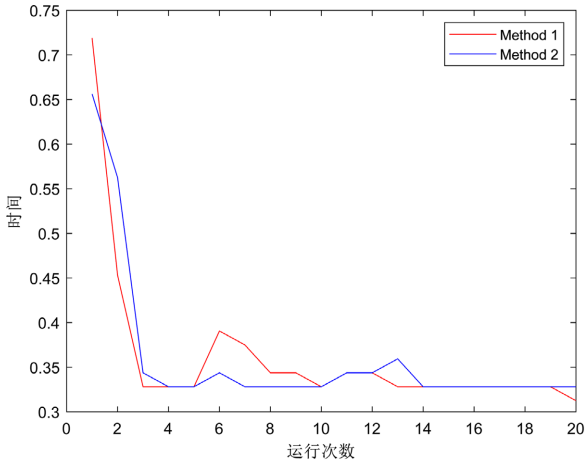
(a) Ackley 函数



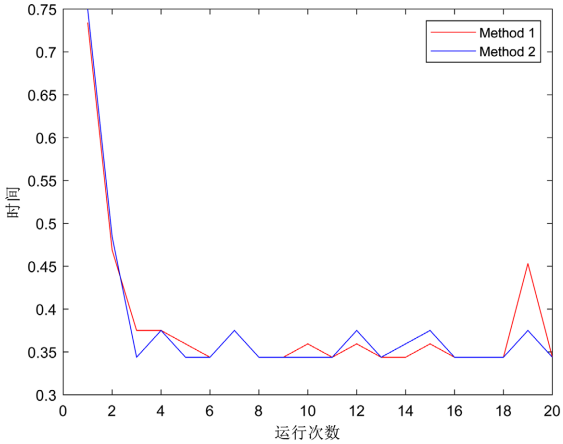
(b) Griewank 函数



(c) Rastrigin 函数



(d) Schaffer 函数



(e) Sphere 函数

Figure 3. Computation time of the two methods
图 3. 两种方式的计算耗时

4.4. 使用控制变量法探索特征值和特征向量对算法搜索性能的影响

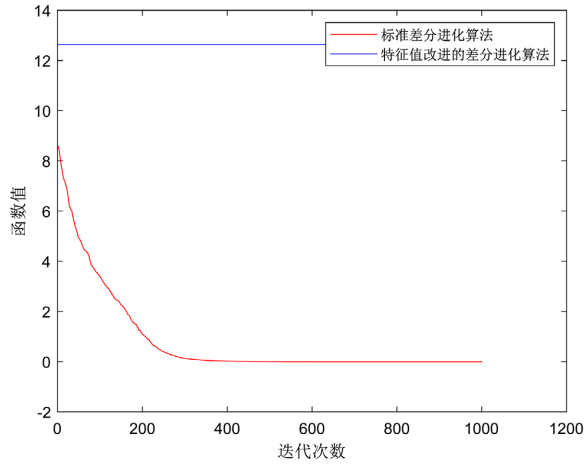
控制变量法是在蒙特卡洛方法中用于减少方差的一种技术方法，把多个因素影响的待解决的问题变成多个单一因素影响的问题；在研究和解决问题时，对影响事物变化规律的因素加以人为控制，只改变其中的某个因素，而控制其余的因素不变，从而研究被改变的因素对事物的影响，分别加以研究，再综合分析，最终解决所研究的问题，控制变量法是科学探究中的重要思想方法，被广泛地运用在各个领域的科研实验中。

在本文研究中，使用控制变量法来研究特征值和特征向量对算法的影响。

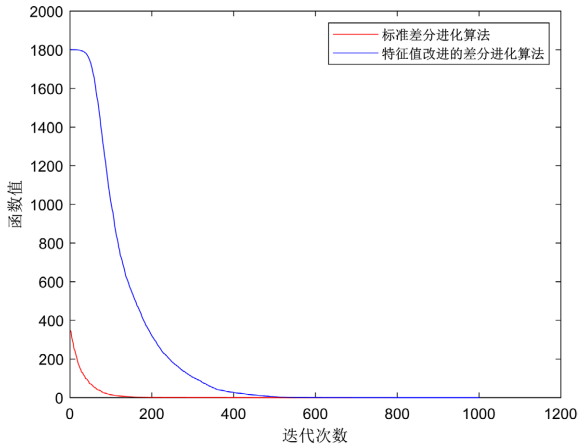
4.4.1. 控制变异个体遵循标准差分进化算法的计算规则

首先控制变异个体的计算规则按照标准的差分进化算法的计算规则来计算，让初始值的计算规则按照使用特征值改进的计算规则来计算，控制特征值的取值来探索协方差矩阵的特征值对算法搜索性能的影响。在标准差分进化算法的程序上做稍些改动，参数设置保留与标准算法一样，把初始值计算规则中的变量上下限差替换成协方差矩阵的特征值，如式(12)所示，此特征值是从特征值矩阵中随机选取的一个值，然后作图与标准的差分进化算法做比较，分析特征值对算法搜索性能的影响。

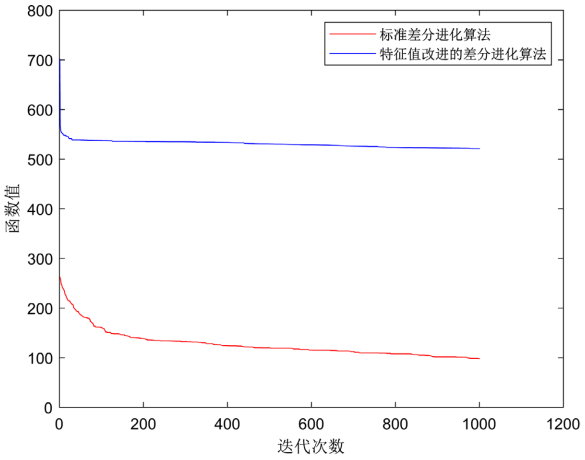
按照上述步骤调试程序，独立测试函数，输出使用特征值改进后的差分进化算法和标准差分进化算法计算得出的函数收敛曲线，如图 4 所示：



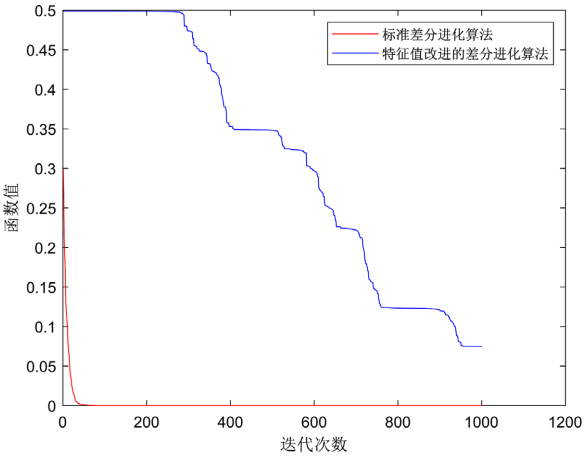
(a) Ackley 函数



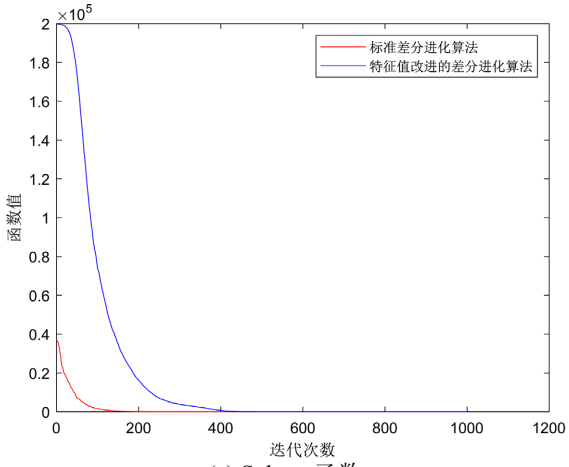
(b) Griewank 函数



(c) Rastrigin 函数



(d) Schaffer 函数



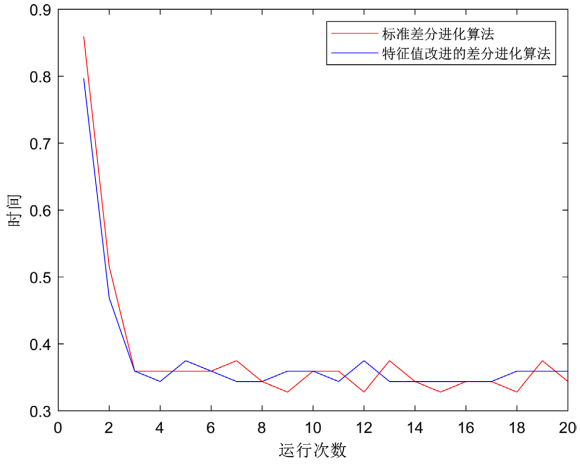
(e) Sphere 函数

Figure 4. Convergence curve of improved eigenvalue algorithm
图 4. 特征值改进算法的收敛曲线

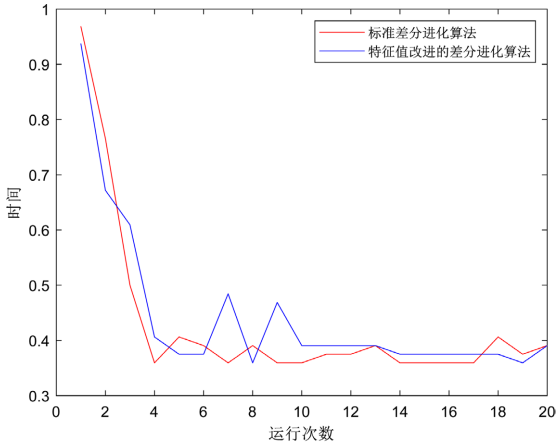
在图 4 中，红色曲线和蓝色曲线分别表示标准的差分进化算法和特征值改进后的差分进化算法对应的计算结果。观察实验结果，使用特征值改进后的差分进化算法在计算测试函数过程中，均提高了函数

收敛初始值，收敛速度也相较于使用标准差分进化算法的收敛速度慢，且易使算法陷入局部最优解。因此，使用特征值改进的差分进化算法未能有效地提高算法搜索性能。

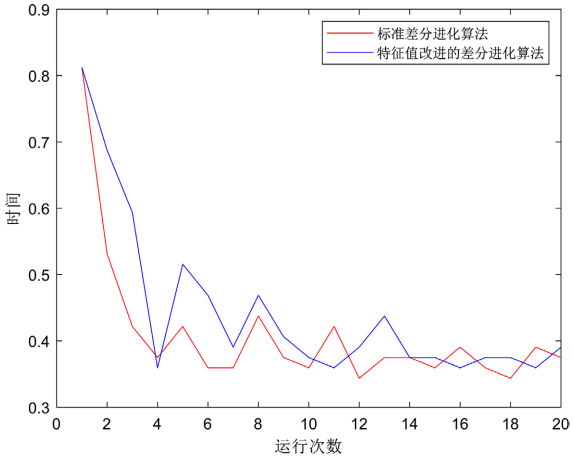
再把使用特征值改进后的差分进化算法和标准差分进化算法运算的耗时作比较，如图 5 所示：



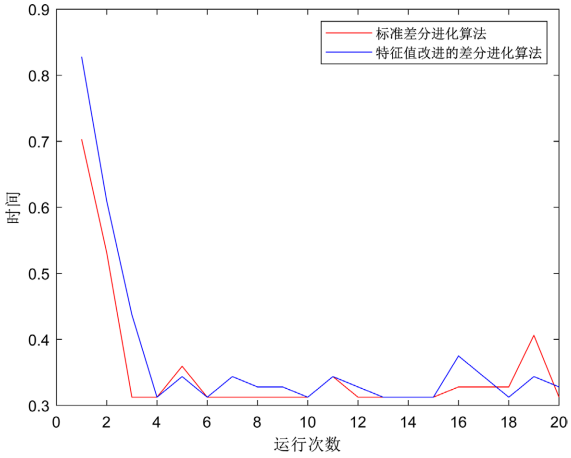
(a) Ackley 函数



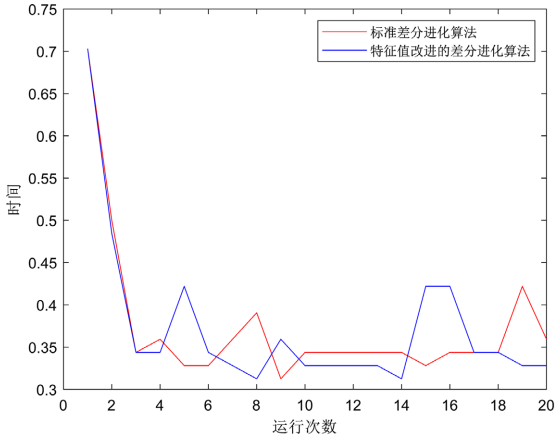
(b) Griewank 函数



(c) Rastrigin 函数



(d) Schaffer 函数



(e) Sphere 函数

Figure 5. Computation time of improved eigenvalue algorithm
图 5. 特征值改进算法的计算耗时

观察图 5 中的实验结果，发现两条曲线有多次相交，且波动幅度较小，说明两个算法在运算时的耗时相近，同时也说明特征值对差分进化算法的运算耗时影响较小。

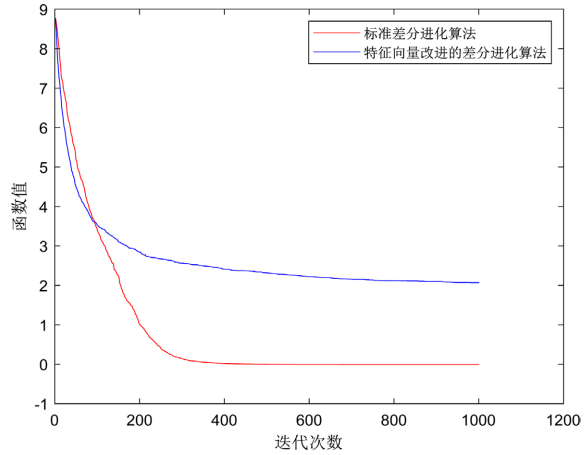
根据上述实验分析，可得出结论如下：按照特征值代替变量上下限差值$(Xs-Xx)$的计算规则来计算初始种群，不仅会使差分进化算法进入更大的局部最优解范围，陷入早熟收敛问题，而且还降低了差分进化算法的收敛速度，因此特征值的此计算规则未能有效地提高差分进化算法的搜索性能。

4.4.2. 控制初始种群遵循标准差分进化算法的计算规则

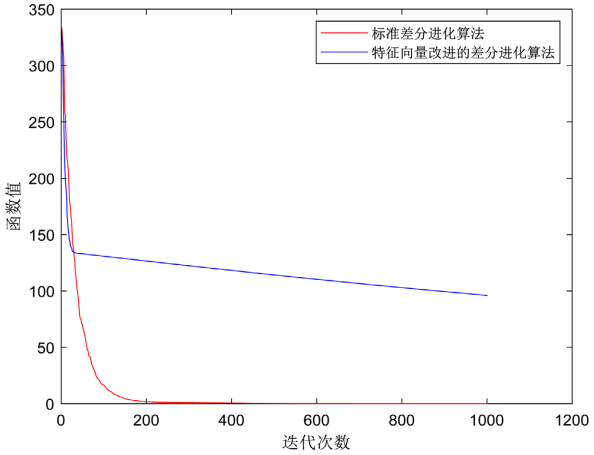
接下来研究的是协方差矩阵的特征向量对算法搜索性能的影响，让初始种群的计算规则遵循标准差分进化算法的初始种群的计算规则，而变异个体的计算规则使用特征向量来加以改进。算法中的参数保持跟标准差分进化算法的参数设置一样，变异个体的计算规则稍作改变，如式(13)所示，从特征向量矩阵中随机抽取一个特征向量代替父代个体中随机选取的两个个体的向量差，研究其对算法搜索性能的影响。

按照上述步骤调试程序，独立测试函数，输出使用特征向量改进后的差分进化算法和标准差分进化算法计算得出的函数收敛曲线，如图 6 所示：

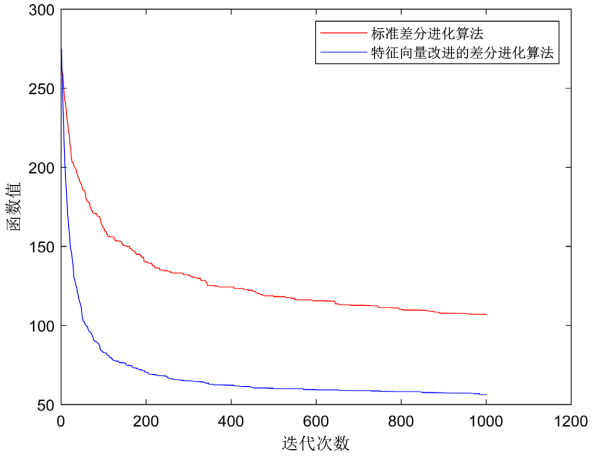
在图 6 中，红色曲线和蓝色曲线分别表示标准的差分进化算法和特征向量改进后的差分进化算法对应的计算结果。观察实验结果，在 Ackley、Griewank、Schaffer 和 Sphere 函数收敛曲线中，刚开始进化时，使用特征向量改进过后的差分进化算法收敛速度明显比标准差分进化算法的收敛速度快，但随着进化次数的增加，函数收敛速度变慢，未能在进化范围内收敛至全局最优解。而在 Rastrigin 函数收敛曲线



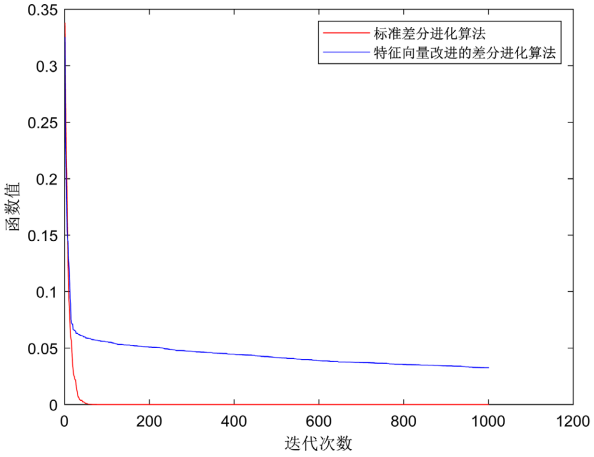
(a) Ackley 函数



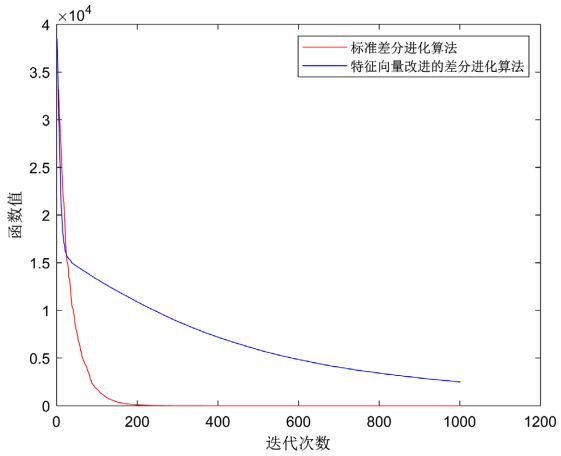
(b) Griewank 函数



(c) Rastrigin 函数



(d) Schaffer 函数

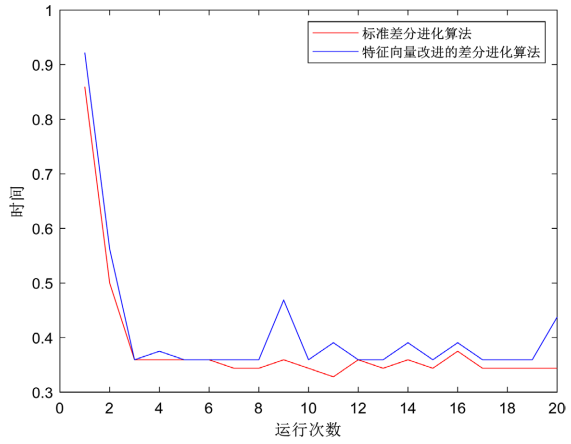


(e) Sphere 函数

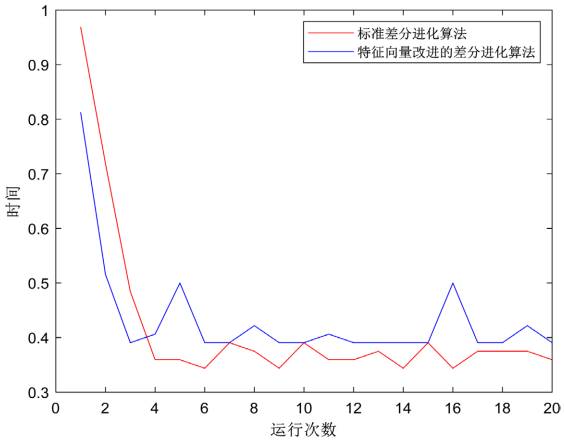
Figure 6. Convergence curve of improved eigenvector algorithm
图 6. 特征向量改进算法的收敛曲线

中，使用特征向量改进后的差分进化算法寻找全局最优解的效果明显优于标准差分进化算法的寻优效果，不仅在收敛速度上，还是函数收敛曲线向下的收敛趋势上，均起到改进的效果。

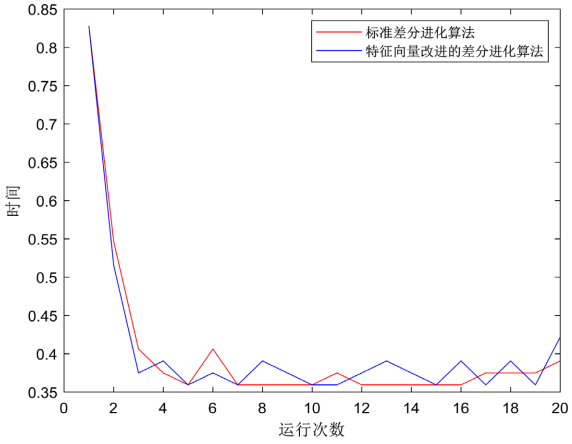
再把使用特征向量改进后的差分进化算法和标准差分进化算法运算的耗时作比较，如图 7 所示：



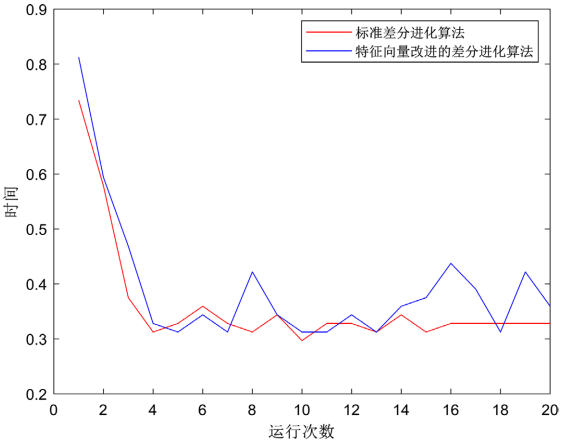
(a) Ackley 函数



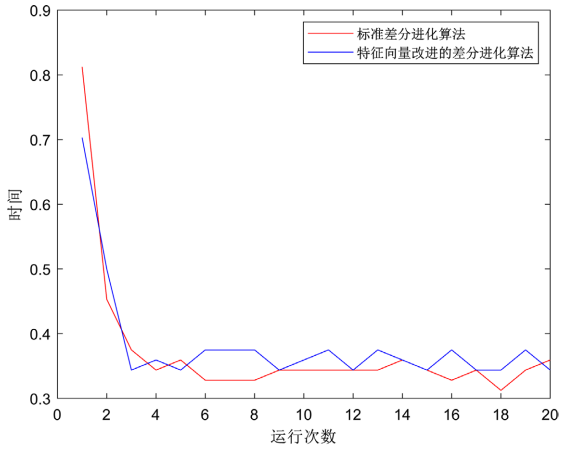
(b) Griewank 函数



(c) Rastrigin 函数



(d) Schaffer 函数



(e) Sphere 函数

Figure 7. The computation time of improved eigenvector algorithm
图 7. 特征向量改进算法的计算耗时

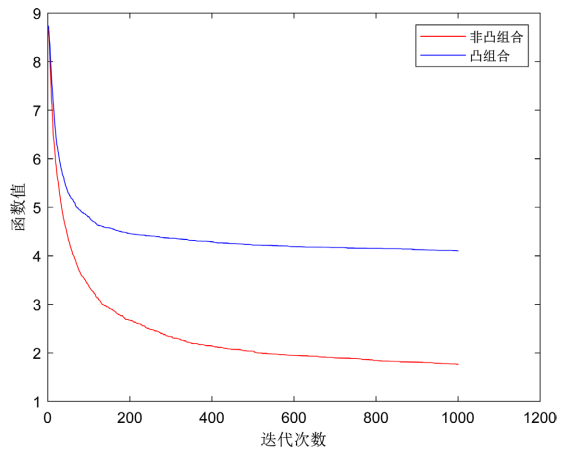
观察图 7 中的实验结果，发现两条曲线相交向前，与使用特征值改进算法实验结果相似，使用特征向量改进的差分进化算法对运算耗时影响较小。

考虑到随机选取特征向量代入变异个体的计算规则计算得出的函数收敛曲线后半段收敛速度减慢的现象，可能是算法在寻优过程中，随着进化次数的增加，导致种群多样性的迅速降低，使得种群中的个体相识度较高，从而使得算法在寻优过程中难度增加，以致于收敛速度变慢。现提出一个想法，从特征向量的矩阵中随机选取三个特征向量，分别代入变异个体的计算规则，得出三个变异个体，然后将这三个变异个体进行凸组合，得到一个最终代入交叉操作的变异个体，以这样的线性组合方式提高变异个体的多样性，从而提高算法中的种群多样性。由于凸组合对系数的要求是非负实数且所以系数之和等于 1，所以在算法程序中如式(16)和式(17)定义凸组合的系数：

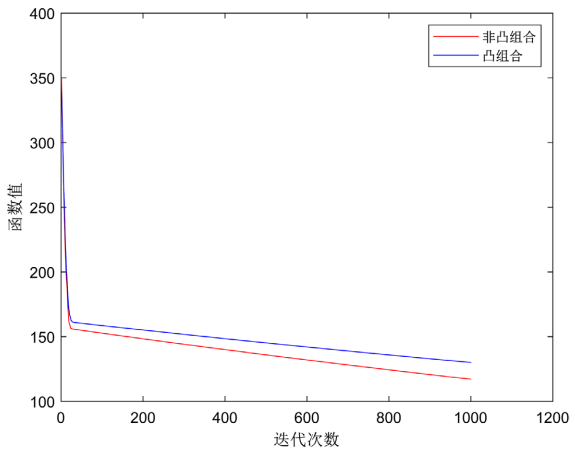
$$w_i = rand(\,) \tag{16}$$

$$ww_i = w_i \Big/ \sum_{i=1}^{3} w_i \tag{17}$$

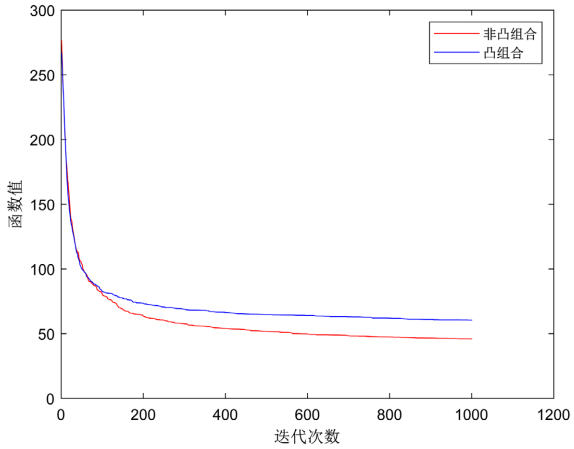
其中，$i = 1,2,3$，ww_i 为凸组合的系数。按照上述想法，分别独立计算测试函数，输出随机选取单个特征向量(非凸组合)改进的差分进化算法和凸组合改进的差分进化算法计算得出的函数收敛曲线，如图 8 所示：



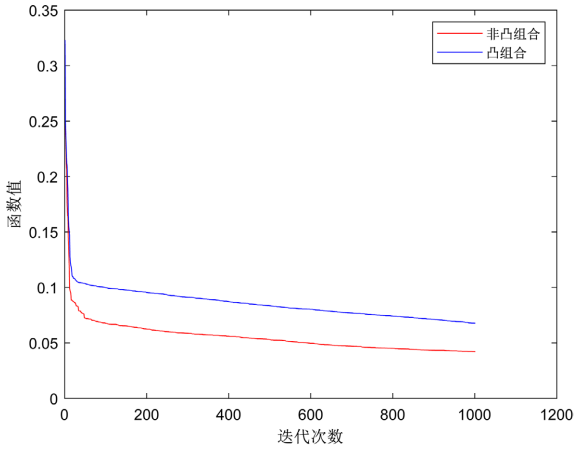
(a) Ackley 函数



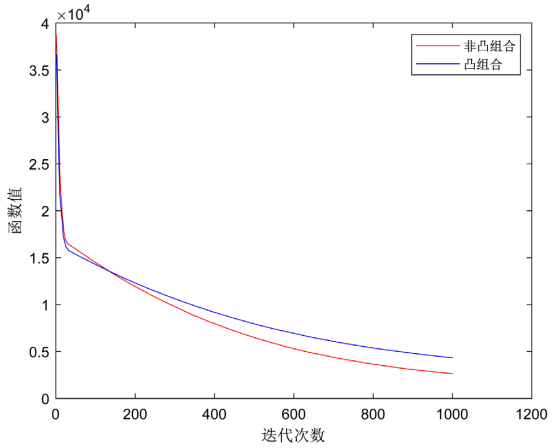
(b) Griewank 函数



(c) Rastrigin 函数



(d) Schaffer 函数



(e) Sphere 函数

Figure 8. Convergence curves of non convex combination and convex combination improved algorithms
图 8. 非凸组合和凸组合改进算法的收敛曲线

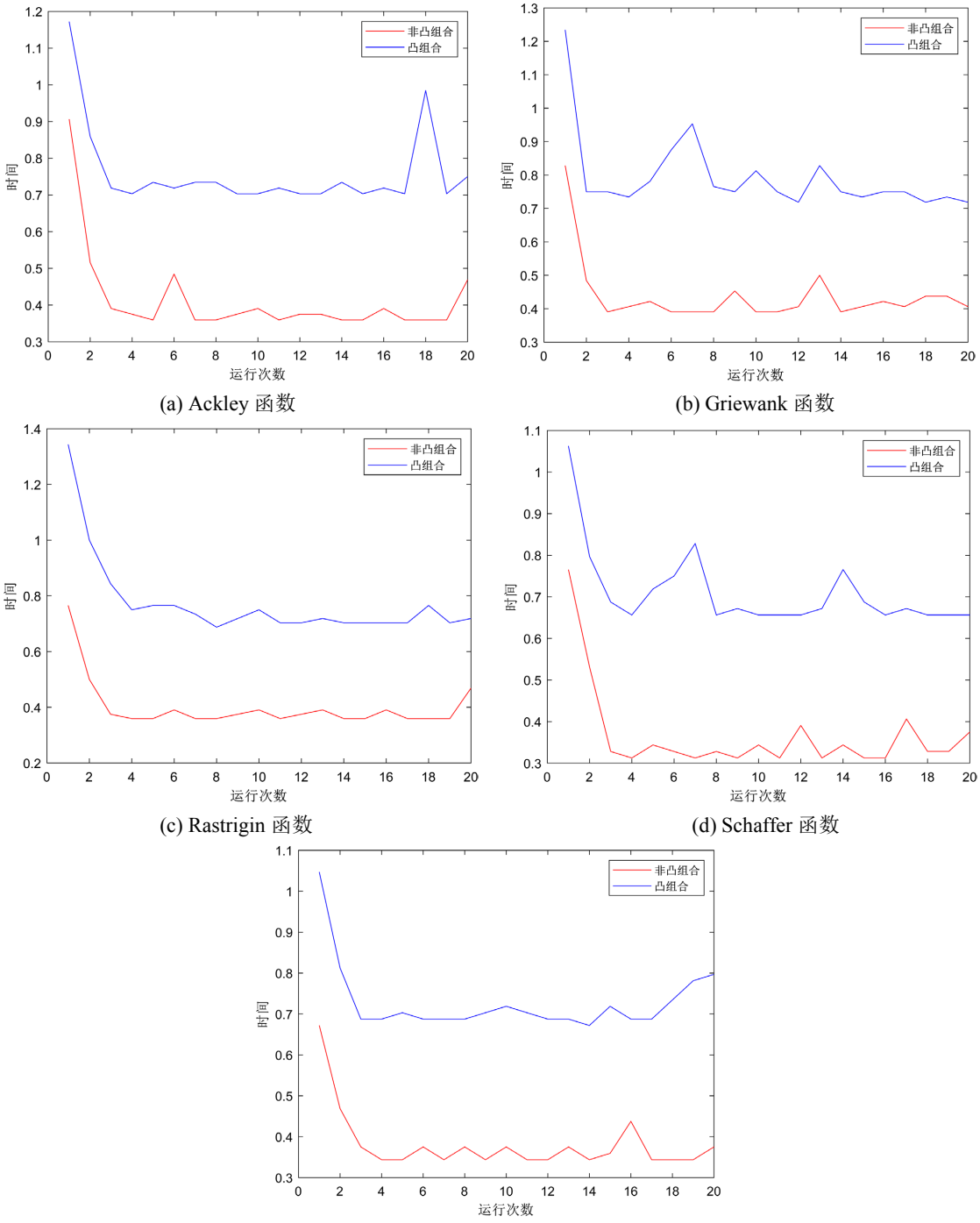
在图 8 中，红色曲线和蓝色曲线分别表示随机选取单个特征向量(非凸组合)改进的差分进化算法和凸组合改进的差分进化算法对应的计算结果。观察实验结果，在 Ackley、Griewank、Rastrigin 和 Schaffer 函数收敛曲线中，使用随机选取单个特征向量(非凸组合)改进的差分进化算法收敛速度及收敛效果优于使用凸组合改进的差分进化算法；在 Sphere 函数收敛曲线中，在刚开始进化中，使用凸组合改进的差分进化算法收敛速度及收敛效果优于使用随机选取单个特征向量(非凸组合)改进的差分进化算法，但随着迭代次数的增加，使用随机选取单个特征向量(非凸组合)改进的差分进化算法收敛速度反超使用凸组合改进的差分进化算法，且保持良好的收敛趋势。通过实验证明，采用凸组合改进的差分进化算法未能解决函数收敛曲线后半段收敛速度减慢的问题。

再把随机选取单个特征向量(非凸组合)改进的差分进化算法和凸组合改进的差分进化算法的运算耗时作比较分析，如图 9 所示：

观察图 9，蓝色曲线均高于红色曲线，说明使用凸组合改进的差分进化算法在计算目标函数时需要耗费更多的时间。

根据设计仿真实验，探究了协方差矩阵的特征向量对算法搜索性能的影响，协方差矩阵的特征向量主要用于改变变异个体的计算规则。在“控制初始种群遵循标准差分进化算法的计算规则”的仿真实验中，涉及到特征向量改变变异个体的两种方式，一种是从特征向量矩阵中随机选取一个特征向量代替父

代个体中随机选取的两个个体的向量差，另一种是随机选取三个特征向量计算变异个体，然后把三个变异个体进行凸组合。两种方式在一定程度上均能提高差分进化算法寻优时的收敛速度，但同样也存在不足之处，就是随着进化次数的增加，算法寻优的收敛速度迅速降低，且未能在进化范围内收敛至全局最优解；另外采用凸组合改变变异个体的方式，使得算法在运算时的耗时增加。



(a) Ackley 函数　(b) Griewank 函数

(c) Rastrigin 函数　(d) Schaffer 函数

(e) Sphere 函数

Figure 9. The computation time of non convex combination and convex combination improved algorithms
图 9. 非凸组合和凸组合改进算法的计算耗时

5. 实验结果分析

通过仿真实验发现，使用特征值代替变量上下限差值$(Xs-Xx)$改变初始种群的计算规则，不仅会使差分进化算法在计算目标函数时进入更大的局部最优解范围，容易陷入早熟收敛问题，而且还降低差分进化算法全局寻优的收敛速度，因此使用特征值代替变量上下限差值$(Xs-Xx)$改变初始种群的计算规则未能有效地提高差分进化算法的搜索性能。

在探究协方差矩阵的特征向量对算法搜索性能影响的实验中，从特征向量矩阵中随机选取一个特征向量代替父代个体中随机选取两个个体的向量差的改进想法在计算 Ackley、Griewank、Schaffer 和 Sphere 函数中，进化初期，目标函数的收敛速度相比于标准差分进化算法的收敛速度有明显提高，但随着进化次数增加，使得种群多样性迅速降低，种群中个体相似度提高，从而使得目标函数的收敛速度锐减，以致于算法不能在进化范围内收敛至全局最优解；而在计算 Rastrigin 函数中，改进算法寻优效果优于标准差分进化算法，其收敛速度比标准差分进化算法寻优的收敛速度快，且在进化范围内寻找到的局部最优解也优于标准差分进化算法寻到的局部最优解。但此改进想法在计算目标函数全局最优解过程中，暴露出进化后半程收敛速度锐减的问题，原因是差分进化算法在寻优过程中，随着进化代数的增加，会使种群多样性降低，种群中的个体相似度提高，会让算法过早的收敛至局部极小值，或者会令算法搜索停滞。因此提出凸组合改进变异个体计算规则的想法。

使用凸组合改进变异个体的计算规则，通过增加变异个体的随机性、多样性，试图以此降低种群的多样性在算法寻优过程中降低的速度。但通过仿真实验发现，除在 Sphere 函数的进化初期的收敛速度有所提高外，其他 4 个测试函数均未达到提高收敛速度和收敛效果的作用，且凸组合改进的差分进化算法在计算目标函数时需要耗费更多的时间。因此凸组合改进变异个体计算规则的想法未能解决算法在进化后半程收敛速度锐减的问题。

整体来说，本文中提出的特征值改进差分进化算法的想法未能有效地提高差分进化算法的搜索性能，反而起到反作用；使用特征向量改进差分进化算法的想法虽未能在仿真实验的进化代数收敛至全局最优解，但在一定程度提高了差分进化算法的收敛速度，在算法寻优初期也能正确的引导算法进行搜索，但随着进化代数的增加，仍存在收敛速度降低使得算法过早收敛的问题。

6. 结语

本文使用协方差矩阵和差分进化算法结合的想法，试图通过改变初始种群和变异个体的计算规则来提高算法的搜索性能；通过仿真实验发现，虽在一定程度上对算法搜索性能有所提高，但算法仍容易陷入局部最优，从而过早收敛。

下一步，在差分进化算法改进过程中，将持续优化初始种群的计算规则，使用差异性较大的元素组成的矩阵用以生成协方差矩阵；除此外，也同时考虑提高种群多样性问题，尝试在算法参数设置中提高种群的数量和控制交叉概率和缩放因子的取值或者是改变变异个体的计算规则来影响种群的多样性；另外，将尝试多种群的差分进化算法，通过增强种群间个体的差异性。

基金项目

《基于重点对象特征及行为模式分析的“智慧监护”平台研究及体系应用》，2020 年度贵阳市国家创新城市“百城百园”行动项目，贵阳市科技局(筑科项目[2020] 22 号)。

参考文献

[1] 张庆科. 粒子群优化算法及差分进化算法研究[D]: [博士学位论文]. 济南: 山东大学, 2017.

[2] 杜永兆, 范宇凌, 柳培忠, 等. 多种群协方差学习进化算法[J]. 电子与信息学报, 2019, 41(1): 1-8.

[3] 丁青锋, 尹晓宇. 差分进化算法综述[J]. 智能系统学报, 2017, 12(4): 431-442.

[4] 沈鑫, 邹德旋, 张鑫. 随机自适应差分进化算法[J]. 电子科技, 2018, 31(2): 51-55.

[5] 刘志军, 唐柳, 刘克铜, 等. 差分演化算法中变异策略的改进与算法的优化[J]. 化工自动化及仪表, 2010, 37(9): 5-8.

[6] 侯莹, 韩红桂, 乔俊飞. 基于参数动态调整的多目标差分进化算法[J]. 控制与决策, 2017, 32(11): 1985-1990.

[7] 马永杰, 朱琳, 田福泽. 动态参数调整的多策略差分进化算法[J]. 西北师范大学学报(自然科学版), 2018, 54(3): 40-46.

[8] 曾辰子, 余旌胡, 邹桢苹. 基于多样变异随机搜索的差分进化算法[J]. 武汉大学学报(理学版), 2018, 64(3): 211-216.

[9] 刘龙龙, 颜七笙. 差分进化算法的改进及其应用研究[J]. 江西科学, 2018, 36(4): 573-578.

[10] Price, K.V., Storn, R.M. and Lampinen, J.A. (2017) Differential Evolution: A Practical Approach to Global Optimization. China Machine Press, Beijing, 1-369.

[11] 朱林波. 差分进化算法和群集蜘蛛优化算法的研究[D]: [硕士学位论文]. 合肥: 安徽大学, 2017.

[12] Qin, A.K., Huang, V.L. and Suganthan, P.N. (2009) Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Transactions on Evolutionary Computation*, **13**, 398-417. https://doi.org/10.1109/TEVC.2008.927706

[13] Igel, C., Hansen, N. and Roth, S. (2007) Covariance Matrix Adaptation for Multi-Objective Optimization. *Evolutionary Computation*, **15**, 1-28. https://doi.org/10.1162/evco.2007.15.1.1

[14] Wairojjana, N., Pakkaranang, N., Uddin, I., *et al.* (2020) Modified Proximal Point Algorithms Involving Convex Combination Technique for Solving Minimization Problems with Convergence Analysis. *Optimization*: *A Journal of Mathematical Programming and Operations Research*, **69**, 1655-1680. https://doi.org/10.1080/02331934.2019.1657115