基于协方差分析的合作协同进化差分进化算法

王彬[1,2]，任露[1]，王晓帆[1]，曹雅娟[1]

（1. 西安理工大学计算机科学与工程学院，陕西 西安 710048；
2. 西安理工大学陕西省网络计算与安全技术重点实验室，陕西 西安 710048）

摘 要：在大规模高维优化问题中，随着决策变量数目的增加，协同进化算法在搜索全局最优解过程中容易陷入局部最优。基于此，提出了一种基于协方差分析的合作协同进化差分进化算法，在根据决策变量之间的相关性对优化问题进行分组之后，针对子组件内部变量之间的相关性会影响种群进化过程的现象，在对子组件优化的过程中，利用协方差计算种群分布的特征向量，通过坐标旋转消除变量之间的相关性，有效避免在种群搜索过程中陷入局部最优，同时加快了算法的寻优速度。在 CEC 2014 测试函数集上进行了对比实验，实验结果表明，所提算法具有可行性。

关键词：大规模优化问题；合作协同进化；相关性；协方差分析；差分进化

中图分类号：TP18

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2023005

Cooperative coevolution algorithm with covariance analysis for differential evolution

WANG Bin[1,2], REN Lu[1], WANG Xiaofan[1], CAO Yajuan[1]

1. Faculty of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China
2. Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an University of Technology, Xi'an 710048, China

Abstract: With the increase of the number of decision variables, cooperative coevolution algorithm is easy to fall into local optimization in the process of searching the global optimal solution in large-scale high-dimensional optimization problems. Based on this, a cooperative coevolution algorithm with covariance analysis for differential evolution was proposed. After the optimization problems were grouped according to the correlation between the decision variables, the correlation between the internal variables of the subcomponents would affect the population evolution process. In the process of subcomponent optimization, covariance was used to calculate the characteristic vector of population distribution, and the correlation between variables was eliminated through coordinate rotation, which effectively avoided falling into local optimization in the process of population search and speeded up the optimization speed of the algorithm. Comparative experiments were carried out on the CEC 2014 test suite. The experimental results show that the proposed algorithm is feasible.

Keywords: large-scale optimization problem, cooperative coevolution, correlation, covariance analysis, differential evolution

0 引言

随着现代科技的不断发展，为了得到最佳的解决方案，需要对工程问题进行数学建模求解，但是许多大规模优化问题在建模之后目标函数不可导，导致传统的优化算法难以求解。为了更好地求解大

收稿日期：2022−07−28；**修回日期：**2022−10−24
通信作者：王彬，wb@xaut.edu.cn
基金项目：国家自然科学基金资助项目（No.61976177, No.U21A20524）
Foundation Item: The National Natural Science Foundation of China (No.61976177, No.U21A20524)

规模优化问题，研究者不断探索并设计出更加高效的进化算法[1-10]。求解大规模优化问题有 2 种主流进化方法，具体介绍如下。

1) 基于非分解的方法

文献[11]提出了多轨道搜索算法，该算法有 3 个全局搜索能力和局部搜索能力不同的搜索算子，在进化过程中，使用这 3 个搜索算子对目标空间进行搜索，3 个搜索算子在种群进化过程中相辅相成，使全局搜索和局部搜索能够得到均衡。文献[12]提出了多后代采样算法，具体是在种群进化的开始为每种算法分配相同的计算资源，在之后的进化过程中，根据种群中个体的适应度值对每种算法进行评价，然后计算出 2 种算法在进化过程中的参与率，根据参与率计算出下一轮优化过程中每种算法被分配的计算资源，不断进化直到计算资源用尽。与其他算法相比，其性能较好，但是多后代采样算法对优化算子的选择比较敏锐，不同优化算子会影响算法的性能。

2) 基于分解的方法

比较典型的是合作协同进化（CC, cooperative coevolution）算法，即对优化问题进行分解，将高维优化问题分解为多个低维优化问题，然后对每个子组件进行优化。合作协同进化算法的性能取决于分解策略，目前研究者已经提出了多种分解策略。文献[13]为了提高遗传算法的性能，首次提出将问题进行分解，使用的分解策略是单维分解和折半分解。单维分解是将 M 维问题分解成 M 个一维问题；折半分解是将整个优化问题一分为二。在 30 维测试函数上进行测试，对于可分的问题，该算法的性能比遗传算法好；对于不可分的问题，由于在分解过程中 2 种策略并没有考虑变量之间的相关性，因此算法性能不如遗传算法好，并且随着优化问题维度的增加，这 2 种分解策略会失效。文献[14]将 CC 框架应用到粒子群优化算法，使用的分解策略是固定分组策略，它将一个 M 维的优化问题分解成 S 个 k 维的问题，即 M=Sk，但是这种固定分组策略只对低于 30 维的优化问题有效。以上几种分解策略都需要预先设置问题分解的数目，且对高维问题效果不佳。文献[15]提出了随机分组策略，可以对变量进行随机分组，这种分组方法增大了将有关联性的变量分到同一子组件的概率，但是随着有关联性变量的数目增加，随机分组的性能就会下降。文献[16]提出了多层次的协同进化算法用来解决上述问题，设置了一个分解器池，里面不同的分解器代表不同的分组数目，记录每个分解器的性能，在进化过程中根据记录的历史数据自适应地选择合适的分解器，根据分解器中的分组数目，将目标向量分解成设定的子组件。虽然多层次的协同进化克服了随机分组手动设置分组大小的缺点，但是在实际应用中并不广泛。文献[17]提出了差分分组（DG, differential grouping）策略，能够检测变量之间的关联性。DG 与其他的分组策略相比具有良好的性能，但它只能检测变量之间的直接相关性，并不会检测间接相关性，在一些测试集上的效果并不理想。文献[18]提出了扩展的差分分组（XDG, extended differential grouping）策略来解决 DG 的缺陷。

总之，合作协同进化算法利用分组策略将大规模优化问题进行分解，加快了进化过程的收敛速度，但当决策变量部分可分离或完全不可分时，协同进化的优化效果并不理想，因为合作协同进化只是根据变量之间的相关性将变量进行一个大的分类，并没有考虑分类之后每个子组件中变量之间的相互作用是否影响进化过程。基于上述分析，本文提出了一种基于协方差分析的合作协同进化差分进化算法，简称 CC-COV-DE 算法。

本文的主要研究工作如下。

1) 在协同进化利用分组策略对决策变量进行分组之后，针对子组件中变量之间的相关性对种群进化过程的影响进行了分析。

2) 在种群进化开始时对问题进行传统分解，在优化每个子组件时利用协方差分析子组件中变量的数据特征，构造协方差矩阵，根据特征向量旋转原始坐标系，目的是消除组内变量之间的相关性，提高算法的性能。

3) 提出了一种基于协方差分析的合作协同差分进化算法，并在 CEC 2014 测试集上与最新的差分进化算法进行了仿真实验，实验结果证实了所提算法的有效性、高效性以及可竞争性。

1 相关工作

1.1 差分进化

进化算法作为元启发式的算法，其思想类似于达尔文进化论，简单易懂，操作容易，可被广泛应用于求解优化问题。差分进化（DE, differential evolution）是经典的进化算法[19]。

差分进化的操作包括突变、交叉和选择，使用

一组实参向量 $\boldsymbol{x}_i=(x_{i1},\cdots,x_{iD}),i=1,2,\cdots,N$ 表示，其中，D 是问题的维度，N 是种群规模。在种群开始进化之前，随机初始化种群中的个体；在进化过程中，通过突变操作产生突变向量，根据交叉操作产生实验向量，最后通过选择操作为下一代选择适应度值较好的个体。差分进化的操作过程如下。

1) 突变

在进化过程中，通过目标向量 $\boldsymbol{x}_{i,G}$ 来产生突变向量 $\boldsymbol{v}_{i,G}$。突变计算式为

$$\boldsymbol{v}_{i,G}=\boldsymbol{x}_{r_1,G}+F(\boldsymbol{x}_{r_2,G}-\boldsymbol{x}_{r_3,G}) \tag{1}$$

其中，r_1,r_2,r_3 是从[1,N]中随机选择的不同于 i 的下标，F 是比例因子。

2) 交叉

根据目标向量 $\boldsymbol{x}_{i,G}$ 和突变向量 $\boldsymbol{v}_{i,G}$，生成实验向量 $\boldsymbol{u}_{i,G}$。二项式交叉操作为

$$\boldsymbol{u}_{j,i,G}=\begin{cases}\boldsymbol{v}_{j,i,G}, & \text{rand}(0,1)\leqslant \text{CR} \quad 或 \quad j=j_{\text{rand}}\\ \boldsymbol{x}_{j,i,G}, & \text{其他}\end{cases} \tag{2}$$

其中，rand(0,1) 是 0 和 1 之间的随机数，j_{rand} 是从[1,D]中随机选择的决策变量的下标，CR 是交叉率。

3) 选择

使用贪婪选择，在目标向量 $\boldsymbol{x}_{i,G}$ 和实验向量 $\boldsymbol{u}_{i,G}$ 之间根据目标函数的适应度值选择较好的个体进入下一代，选择计算式为

$$\boldsymbol{x}_{G+1}=\begin{cases}\boldsymbol{u}_{i,G}, & f(\boldsymbol{u}_{i,G})<f(\boldsymbol{x}_{i,G})\\ \boldsymbol{x}_{i,G}, & \text{其他}\end{cases} \tag{3}$$

由 DE 的操作过程可以看出，当控制参数较少时，操作相对简单，可以在低维优化问题中快速找到最优解。

1.2　合作协同进化框架

合作协同进化框架最早在 1994 年被提出来解决大规模优化问题，首先将高维问题分解成多个低维问题，然后对子组件进行循环优化。合作协同进化算法的伪代码如算法 1 所示。

算法 1　合作协同进化算法

1) 初始化种群
2) 使用分组策略将原始问题分解为 m 个低维度子组件
3) set $i=1$ 开始一个新循环
4) 使用特定的进化算法来优化第 i 个子组件并分配固定数量的评估时间
5) if $i<m$ then i++，转至步骤 3)
6) end if
7) if 满足停止条件 do
8)　停止循环
9) else
10)　转至步骤 2)进行下一个循环
11) end if

1.3　扩展的差分分组策略

协同进化求解大规模优化问题的核心是如何对问题进行分解，即选择什么样的分解策略。

实际工程领域中的优化问题较复杂，扩展的差分分组策略弥补了差分分组在识别相互关系上的缺陷，XDG 可以识别变量之间的 2 种交互类型，如图 1 所示。类型Ⅰ表示变量之间的直接交互关系，如 x_1 和 x_2、x_2 和 x_3；类型Ⅱ表示变量之间的间接交互关系，如 x_1 和 x_3。

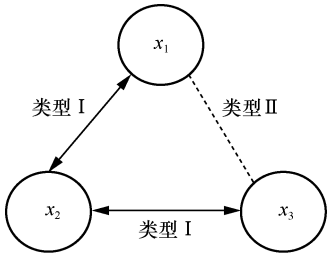

图 1　变量之间的 2 种交互类型

变量之间的相关性定义如下。

定义 1　$f(\boldsymbol{x})$ 是部分可分函数，对于 $\forall a,b_1\neq b_2$，$\delta\in R,\delta\neq 0$，如果满足

$$\triangle_{\delta,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_1}\neq\triangle_{\delta,x_p}[f](\boldsymbol{x})|_{x_p=a,x_q=b_2} \tag{4}$$

则 x_p 和 x_q 不可分离，存在相关性，其中，

$$\triangle_{\delta,x_p}[f](\boldsymbol{x})=f(\cdots,x_p+\delta,\cdots)-f(\cdots,x_p,\cdots) \tag{5}$$

定义 2　在目标函数 $f(\boldsymbol{x})$ 中，如果 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 是直接相关的，则存在候选解 $\boldsymbol{x}_*$ 满足式（6），直接相关定义为 $\boldsymbol{x}_i\leftrightarrow\boldsymbol{x}_j$。

$$\frac{\partial f}{\partial \boldsymbol{x}_i\partial \boldsymbol{x}_j}|_{\boldsymbol{x}_*}\neq 0 \tag{6}$$

如果 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 是间接相关的，则所有的候选解满足式(7)。

$$\frac{\partial f}{\partial \boldsymbol{x}_i\partial \boldsymbol{x}_j}=0 \tag{7}$$

XDG 根据决策变量之间的相互关系将原问题进行分解。在对问题进行分解时，其可分成 3 个阶

段。第一阶段是确定变量之间的直接交互，根据定义 1 以成对的方式检测变量之间的交互关系。第二阶段是为了确定变量之间的间接交互关系，搜索第一阶段中子组件中的重叠部分，合并公共变量的子组件。第三阶段是将所有可分离变量分配到同一子组件中，XDG 分组之后每个子组件之间是可分离的，但是子组件中的变量是相互关联的。

2 算法设计

2.1 研究动机

1) 高维问题中子组件变量相关增加

求解大规模优化问题最常见的方法是将高维问题分解成低维问题，即根据决策变量的相关性对所求解的优化问题，将有相互依赖关系的变量分配到同一个子组件中，然后对所有的子组件循环进行优化。在问题完全可分的情况下，利用这种方法求解优化问题可以很大程度地提高算法的性能，加快种群的收敛。

如图 2 所示，低维优化问题中，由于维度较低，部分子组件内部变量关联性较低的概率较大，其中，子组件 1、子组件 2 和子组件 4 中的变量有相互依赖关系，子组件 3 中的变量在进化过程中不依赖其他任何变量，因而在进化过程中子组件 3 的进化速度要比其他组件的进化速度快。

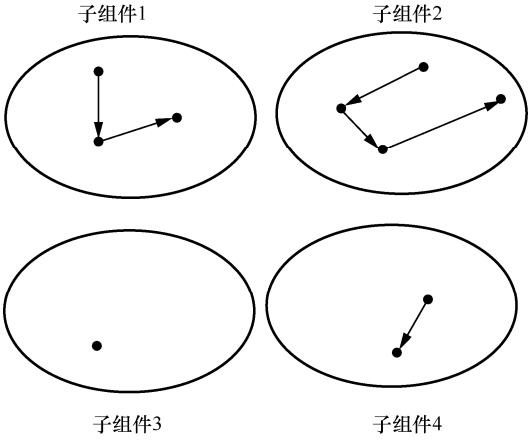

图 2 低维优化问题的变量相关

如图 3 所示，高维优化问题中，随着维度的增加，子组件中被分配到的变量数目增加，相较于图 2 中的低维问题，分组之后，每个子组件中变量之间的关联概率增加。

2) 子组件的变量相关性影响协同进化

协同进化在种群的进化过程中并没有考虑子组件中相互作用的变量对进化的影响，利用差分算法对子组件进行优化的过程中，变量之间的相互性会影响子组件的优化。

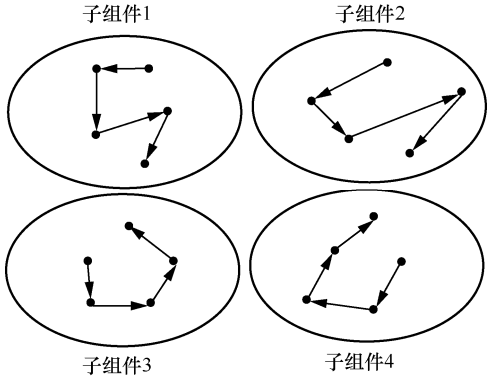

图 3 高维优化问题的变量相关

相关变量对 DE 搜索过程的影响如图 4 所示。理想情况下，根据 DE 的突变计算式在目标空间中产生的突变向量是 $\boldsymbol{v}_{i,G}$，但是向量 $\boldsymbol{x}_{r_1,G}$ 与向量 $\boldsymbol{x}_{r_4,G}$ 是相关的，向量 $\boldsymbol{x}_{r_4,G}$ 的变化会影响向量 $\boldsymbol{x}_{r_1,G}$。在进化过程中，$\boldsymbol{x}_{r_1,G}$ 会受 $\boldsymbol{x}_{r_4,G}$ 的影响，从而产生突变向量 $\boldsymbol{v}_{i,G}$，$\boldsymbol{v}_{i,G}$ 相较于 $\boldsymbol{v}_{i,G}$ 偏离了全局最优向量。

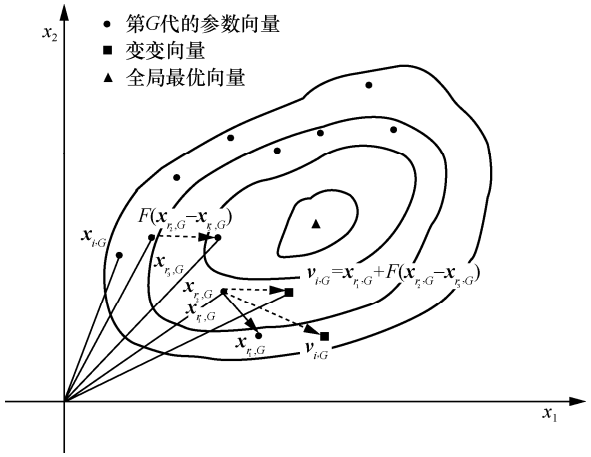

图 4 相关变量对 DE 搜索过程的影响

3) 采用协方差分析的坐标旋转消除变量相关

基于种群个体分布特征向量的坐标旋转如图 5 所示。优秀个体往往分布在最优解周围，在假设高斯分布的基础上，利用优秀个体的位置，计算种群的协方差矩阵，估计分布的特征向量和特征值；利用计算出来的特征向量，对原坐标系（X,Y）进行旋转，并重新计算个体在新坐标系（X',Y'）下的坐标；在新坐标系下，个体的变量相关性降低，进化效率得到提高；将在新坐标系下进化得到的个体映射回原坐标系，计算适用度。

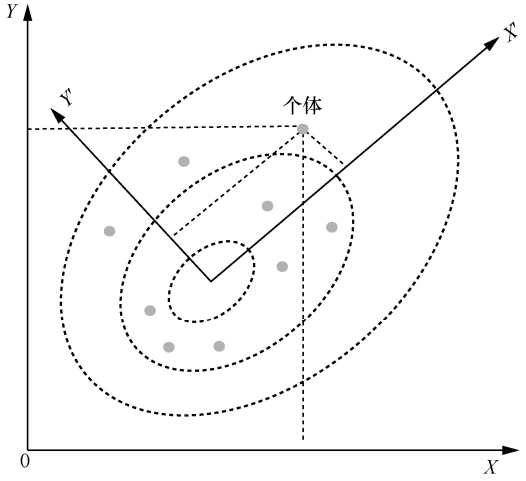

图 5　基于种群个体分布特征向量的坐标旋转

由于当问题规模较大且较复杂时，分组之后每个子组件中有相互作用的变量会增加，从而导致在进化过程中偏离全局最优解。因此，本文在进化过程中利用协方差分析对子组件中变量的相互依赖关系进行消除，加快种群的收敛。

2.2　CC-COV-DE 算法设计

在进化过程中，根据变量之间的相关性对变量进行分组，当子组件中相互依赖的变量数目有很多时，会受变量之间的相互依赖牵引，种群在进化过程中很容易陷入局部最优。因此，为了提高算法的性能，提出了基于协方差分析的合作协同进化差分进化算法，在使用分组策略对决策变量进行分组之后，使用协方差分析每个子组件中变量的特征，根据特征向量对坐标进行旋转，消除变量之间的相关性。

CC-COV-DE 算法的流程如图 6 所示。

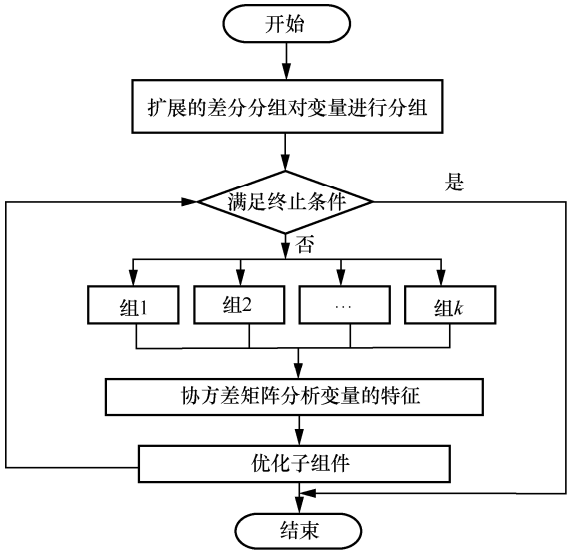

图 6　CC-COV-DE 算法的流程

1) 利用 XDG 策略根据变量之间的相关性对问题进行分解。

2) 在子组件优化过程中，使用协方差对子组件中的变量进行特征分析，构造协方差矩阵，对原始坐标系进行旋转，使子组件中变量之间的关联性可以消除，从而避免在进化过程中陷入局部最优，导致种群进化过程停滞。

在差分进化优化器中，对于子组件中的 M 个个体，计算协方差矩阵

$$\mathrm{cov}\left(P_{1:\frac{M}{2}}\right)=[\mathrm{cov}(i,j)]_{D\times D} \tag{8}$$

其中，$\mathrm{cov}(i,j)$ 是子组件中 M 个个体的第 i 和第 j 维度的协方差，计算式为

$$\mathrm{cov}(k,j)=\frac{1}{\frac{M}{2}-1}\sum_{i=1}^{\frac{M}{2}}\left(x_{i,k}-\overline{x_k}\right)\left(x_{i,j}-\overline{x_j}\right)$$
$$k,j=1,2,\cdots,D \tag{9}$$

$\mathrm{cov}\left(P_{1:\frac{M}{2}}\right)$ 分解为

$$\mathrm{cov}\left(P_{1:\frac{M}{2}}\right)=R\wedge^2 R' \tag{10}$$

其中，R 表示 $D\times D$ 正交矩阵特征坐标系，R 的每一行都是协方差矩阵 $\mathrm{cov}\left(P_{1:\frac{M}{2}}\right)$，$R'$ 表示从特征坐标系到原始坐标系的变换，$\wedge$ 表示由特征值组成的对角矩阵。

在目标空间中，子组件中的个体 $\boldsymbol{x}_i$ 在特征坐标系中可表示为

$$\boldsymbol{x}_i'=\boldsymbol{x}_iR \tag{11}$$

子组件中的个体使用 DE 搜索方程在特征坐标系下生成候选解 $\boldsymbol{v}_j'$

$$\boldsymbol{v}_{i,j}'=\boldsymbol{x}_{r_1,j}'+F(\boldsymbol{x}_{r_2,j}'-\boldsymbol{x}_{r_3,j}') \tag{12}$$

其中，r_1,r_2,r_3 是从[1,M]中随机选择的。将特征坐标系中的候选解转换到原始坐标系中，有

$$\boldsymbol{v}_i=\boldsymbol{v}_i'R' \tag{13}$$

3) 根据目标函数的适应度值，选出最优的个体进入下一代。

CC-COV-DE 算法如算法 2 所示，其中步骤 21)~步骤 29)是协方差分析的过程。

算法 2 CC-COV-DE 算法

输入 种群规模 N，决策变量的维数 D，当前种群的更新代数 gen，比例因子 F，交叉率 CR，函数的最大评价次数 $\mathrm{FES}_{\max}$

输出 评价次数 FES，最优值 Best

1) 初始化种群
2) gen=0;
3) 在搜索空间随机产生 N 个解 $\mathbf{pop}_i(i=1,2,\cdots,N)$
4) 按照目标函数值评价种群
5) 设置当前函数评估 FES = N
6) 使用 XDG 进行分组
7) while FES $\leqslant \mathrm{FES}_{\max}$ do
8) $(\mathrm{best},\mathrm{best\_val}) \leftarrow \min(f(\mathrm{pop}))$;
9) for j=1 to size(groups) do
10) indices $\leftarrow$ groups[j];
11) for i=1 to N do
12) 创建向量 $\boldsymbol{x}_i$ 通过取相应子种群（indices）从 pop_i;%差分变异算子
13) 利用式(1)创建变异向量 $\boldsymbol{v}_i$;%二项式交叉算子
14) 利用式(2)创建实验向量 $\boldsymbol{u}_i$;%贪婪的选择算子
15) 分别替换 $\boldsymbol{u}_i$ 和 $\boldsymbol{x}_i$ 为最佳，评估目标函数值 $f(\boldsymbol{u}_i)$ 和 $f(\boldsymbol{x}_i)$;
16) if $f(\boldsymbol{u}_i) < f(\boldsymbol{x}_i)$ do
17) 替换 $\boldsymbol{u}_i$ 到 $\mathbf{pop}_i$;
18) end if
19) end for
20) 评估个体的目标函数值并对其进行排序;
21) 使用子种群 j 的 M 个最好的个体计算协方差矩阵;
22) for i=1 to N do
23) 利用式(12)产生候选解 $\boldsymbol{v}'_{i,j}$ 在本征坐标系;
24) 利用式(13)转换 $\boldsymbol{v}'_{i,j}$ 到原始坐标系;
25) 替换 $\boldsymbol{v}_i$ 和 $\boldsymbol{x}_i$ 为最好，评估目标函数值 $f(\boldsymbol{v}_i)$ 和 $f(\boldsymbol{x}_i)$
26) if $f(\boldsymbol{v}_i) < f(\boldsymbol{x}_i)$ do
27) 替换 $\boldsymbol{v}_i$ 到 $\mathbf{pop}_i$;
28) end if
29) end for
30) gen = gen + 1;
31) end for
32) end while

3 实验结果分析

为了评估 CC-COV-DE 的性能，本文在 CEC 2014 基准测试函数集进行了仿真实验。对比实验表明 CC-COV-DE 在解决大规模优化问题的算法性能。

3.1 对比算法

为了证明 CC-COV-DE 算法的有效性，本文引入了其他 7 个对比算法，分别是基于合作协同进化的差分（CCDE）算法[20]、基于协方差分析的差分进化（COVDE）算法[21]、差分进化（DE）算法[19]、基于自适应维度水平调整框架的差分进化（ADLADE）算法[3]、基于全局数值优化组合策略的自适应差分进化（CSDE）算法[22]、基于时变策略的差分进化（TVDE）算法[23]、基于合作协同进化和协方差的自适应差分进化（A-CC/COV-DE）算法[24]。

3.2 测试函数

CC-COV-DE 在 CEC 2014 上进行数值实验，为了确保实验结果的准确性，每个算法在测试函数上独立运行 50 次，CEC 2014 的测试函数可以分为 4 类。F1~F3：单峰函数，不可分函数；F4~F16：简单多峰函数，F8 和 F10 为可分函数，其余为不可分函数；F17~F22：混合函数，不可分函数；F23~F30：复合函数，不可分函数。

3.3 参数设置

所有的仿真实验都是在配置 3.40 GHz CPU 和 8 GB RAM、Microsoft Windows 7 操作系统、Intel（R）Core™ i7-3770M 的计算机端进行的，测试软件是 Microsoft Windows 7 操作系统下的 MATLAB 2016a。其中，对比算法中公共参数设置如下。

1) 种群规模：$N=2D$。

2) 测试函数的维度：$D=\dfrac{30}{50}$。

3) 停止准则：为了使算法能够正常停止运行，对于 CEC 2014 测试函数集，函数的最大评价次数（$\mathrm{FES}_{\max}$）设置为 10 000D。

4) 独立运行次数：在本文实验中，设置最大独立运行次数（$\mathrm{run}_{\max}$）为 50 次。

3.4 数值实验

本节对 8 个算法进行了对比实验，利用收敛速度和数值分析对实验结果进行了比较，体现了各个算法的差异。

1) 收敛速度。将 8 个算法在不同测试函数上的收敛情况用曲线绘制出来，能够很直观地看出各个算法求解出的最优解在收敛过程中的误差值。

2) 数值分析。为了更加客观地评价基于协方差分析的合作协同差分进化算法与对比算法之间的性能，本文使用 Wilcoxon 秩和检验（0.05 显著水平）对实验结果进行统计分析，如果得到的 $p>0.05$，则认为比较的 2 个算法没有显著的差异，否则就是有显著的差异。根据 Wilcoxon 秩和检验结果将 CC-COV-DE 与其他对比算法之间的实验结果标记为“+/–/~”，即 CC-COV-DE 与其他算法的结果相比较好、较差或相近。表 1 记录了 8 个算法在 50 维 CEC2014 上的统计结果（运行 50 次）。

从表 1 中可以看出，CC-COV-DE 的实验效果比 CCDE 好，说明在使用 XDG 根据决策变量之间的相关性对问题进行分解之后，子组件中变量的相关性影响种群的进化过程，利用协方差分析子组件中变量的数据特征，根据坐标旋转可以消除变量之间的关联性，提高了算法的性能。从 CC-COV-DE 与其他对比算法的实验结果可以看出，CC-COV-DE 的算法性能整体较

表 1　　8 个算法在 50 维 CEC2014 上的统计结果（运行 50 次）

函数	CCDE	COVDE	DE	ADLADE	CSDE	TVDE	A-CC/COV-DE	CC-COV-DE
F1	8.01+	7.24×10^{5} +	1.43×10^{6} +	1.40×10^{6} +	1.73×10^{6} +	6.70×10^{5} +	2.50×10^{-1} +	1.49×10^{-4}
F2	4.22×10^{6} +	1.63×10^{2} –	3.02–	3.63–	1.53×10^{2} –	6.75×10^{-9} –	3.41×10^{4} –	8.88×10^{5}
F3	1.71×10^{1} +	4.44×10^{-12} –	2.75×10^{-1} +	4.69×10^{-1} ~	3.06×10^{-1} +	6.19×10^{-6} –	1.35×10^{-5} –	1.27×10^{-1}
F4	9.34×10^{1} +	8.24×10^{1} –	5.78×10^{1} –	6.27×10^{1} –	9.66×10^{1} +	8.96×10^{1} +	8.13×10^{1} –	8.80×10^{1}
F5	2.12×10^{1} +	2.09×10^{1} ~	2.11×10^{1} +	2.08×10^{1} -	2.05×10^{1} –	2.07×10^{1} –	2.11×10^{1} +	2.09×10^{1}
F6	3.99–	4.44×10^{1} +	1.42–	1.75–	4.94–	6.14×10^{-1} –	4.13–	5.59
F7	6.79×10^{-1} +	1.36×10^{-13} –	1.27×10^{-13} –	1.48×10^{-4} –	1.91×10^{-13} –	1.82×10^{-13} –	3.80×10^{-1} –	5.42×10^{-1}
F8	9.48×10^{-13} ~	1.37×10^{2} +	1.87×10^{2} +	3.63×10^{1} +	2.03×10^{1} +	1.40×10^{1} +	9.35×10^{-13} ~	9.94×10^{-13}
F9	8.98×10^{1} +	1.63×10^{2} +	3.53×10^{2} +	5.72×10^{1} +	5.59×10^{1} +	4.96×10^{1} +	8.79×10^{1} +	4.28×10^{1}
F10	7.99×10^{-3} ~	6.64×10^{3} +	9.19×10^{3} +	8.56×10^{2} +	9.82×10^{2} +	4.53×10^{2} +	6.50×10^{-3} +	5.00×10^{-3}
F11	4.03×10^{3} +	7.50×10^{3} +	1.30×10^{4} +	4.63×10^{3} +	5.34×10^{3} +	7.26×10^{3} +	4.07×10^{3} +	2.42×10^{3}
F12	3.42+	1.58~	3.27+	1.50~	6.22×10^{-1} –	1.08–	1.76+	1.60
F13	5.50×10^{-1} +	4.02×10^{-1} –	4.61×10^{-1} ~	3.87×10^{-1} –	2.36×10^{-1} –	3.09×10^{-1} –	5.69×10^{-1} +	4.77×10^{-1}
F14	3.62×10^{-1} +	2.61×10^{-1} ~	3.38×10^{-1} +	3.64×10^{-1} +	2.85×10^{-1} +	3.39×10^{-1} +	2.79×10^{-1} ~	2.76×10^{-1}
F15	3.25×10^{1} +	1.88×10^{1} ~	3.10×10^{1} +	2.24×10^{1} +	7.20–	1.29×10^{1} –	1.87×10^{1} ~	1.87×10^{1}
F16	2.23×10^{1} +	2.12×10^{1} ~	2.21×10^{1} +	2.09×10^{1} –	2.02×10^{1} –	1.98×10^{1} –	2.12×10^{1} ~	2.12×10^{1}
F17	7.72×10^{4} +	1.29×10^{3} –	1.58×10^{4} +	1.52×10^{4} +	5.04×10^{4} +	3.44×10^{4} +	1.14×10^{3} –	4.01×10^{3}
F18	1.21×10^{2} +	8.91×10^{1} +	1.35×10^{2} +	1.46×10^{2} +	1.30×10^{2} +	4.03×10^{2} +	3.30×10^{1} –	4.72×10^{1}
F19	6.89–	1.14×10^{1} +	1.22×10^{1} +	1.06×10^{1} +	1.13×10^{1} +	1.74×10^{1} +	6.74–	9.42
F20	7.19×10^{1} +	6.04×10^{1} +	9.81×10^{1} +	1.06×10^{2} +	6.74×10^{1} +	1.02×10^{2} +	3.58×10^{1} +	3.39×10^{1}
F21	1.35×10^{4} +	1.05×10^{3} +	2.64×10^{3} +	2.71×10^{3} +	1.20×10^{4} +	4.24×10^{4} +	7.65×10^{2} –	9.32×10^{2}
F22	1.43×10^{3} +	5.54×10^{2} –	7.66×10^{2} –	1.83×10^{2} –	3.27×10^{2} –	3.78×10^{2} –	6.25×10^{2} –	9.49×10^{2}
F23	3.44×10^{2} ~	3.44×10^{2} ~	3.44×10^{2} ~	3.44×10^{2} ~	3.44×10^{2} ~	3.44×10^{2} ~	3.44×10^{2} ~	3.44×10^{2}
F24	2.70×10^{2} ~	2.70×10^{2} ~	2.70×10^{2} ~	2.70×10^{2} ~	2.62×10^{2} –	2.64×10^{2} –	2.70×10^{2} ~	2.70×10^{2}
F25	2.06×10^{2} +	2.05×10^{2} ~	2.05×10^{2} ~	2.05×10^{2} ~	2.07×10^{2} +	2.08×10^{2} +	2.06×10^{2} ~	2.06×10^{2}
F26	1.01×10^{2} +	1.02×10^{2} ~	1.00×10^{2} +	1.00×10^{2} ~	1.00×10^{2} ~	1.16×10^{2} +	1.06×10^{2} +	1.00×10^{2}
F27	3.78×10^{2} –	7.89×10^{2} ~	3.70×102 –	3.71×10^{2} –	3.47×10^{2} –	3.51×10^{2} –	6.89×10^{2} ~	6.85×10^{2}
F28	1.07×10^{3} –	1.35×10^{3} ~	1.08×10^{3} –	1.08×10^{3} –	1.07×10^{3} –	1.06×10^{3} –	1.36×10^{3} ~	1.38×10^{3}
F29	1.20×10^{3} +	7.05×10^{5} +	9.40×10^{2} +	9.90×10^{2} +	1.60×10^{3} +	1.47×10^{3} +	6.57×10^{2} +	6.16×10^{2}
F30	8.37×10^{3} ~	8.46×10^{3} ~	8.25×10^{3} –	8.36×10^{3} ~	8.30×10^{3} ~	8.59×10^{3} +	8.47×10^{3} ~	8.42×10^{3}
+/–/~	21/4/5	11/7/12	18/8/4	13/10/7	15/12/3	16/13/1	10/10/10	

好，尤其是在优化混合函数和复合函数的效果方面，说明CC-COV-DE算法在优化复杂函数时的性能良好，稳健性强，与最新的算法相比具有可竞争性。

为了能够更加清楚地比较 CC-COV-DE 与对比算法的收敛速度，图 7 和图 8 分别给出了 8 个算法在 30 维和 50 维的测试集上的平均收敛效果。从图 7 和图 8 可以看出，CC-COV-DE 的整体收敛效果优于其他对比算法。

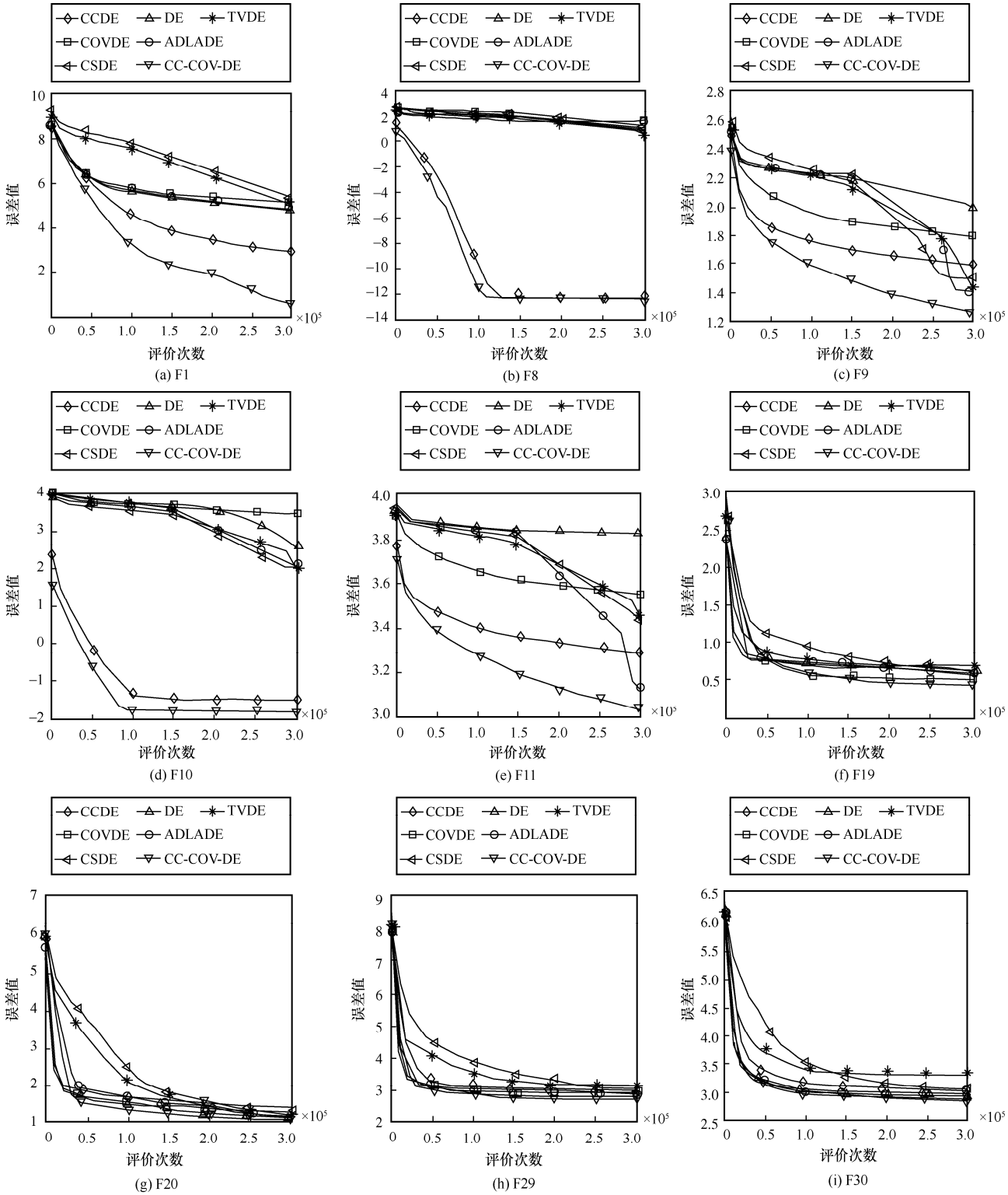

图 7 8 个算法在 30 维的测试集上的部分收敛效果

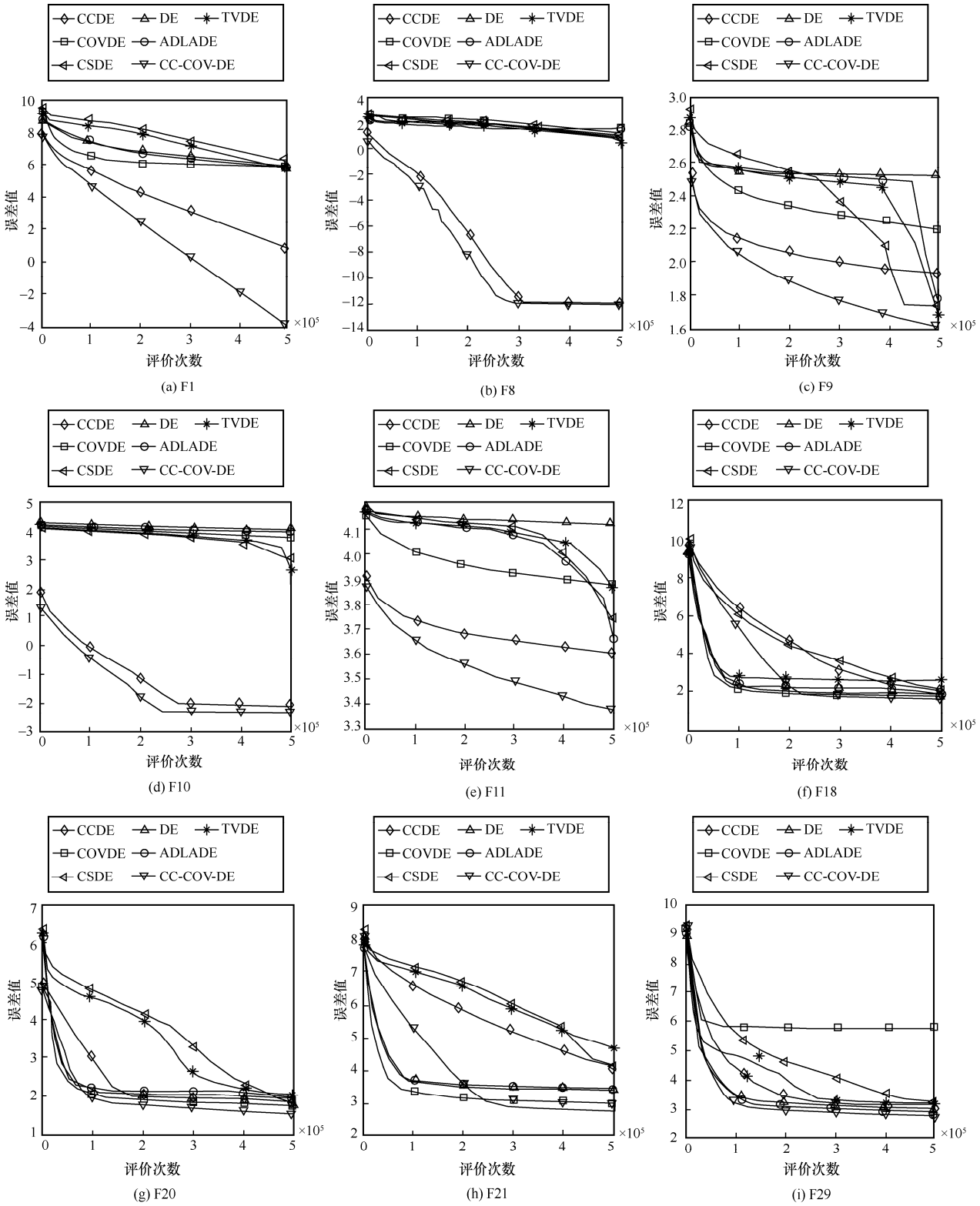

图 8　8 个算法在 50 维的测试集上的平均收敛效果

通过计算 CC-COV-DE 与其他 7 个算法在 30 维的 CEC2014 测试函数集上的运行时间，并且比较其运行时间的比率，分析 CC-COV-DE 的计算效率，实验结果对比如图 9 所示，横线代表运行时间的平均值。在 30 个测试函数中，CC-COV-DE 的平均运行时间分别是 CCDE、COVDE、DE、ADLADE、CSDE、TVDE、

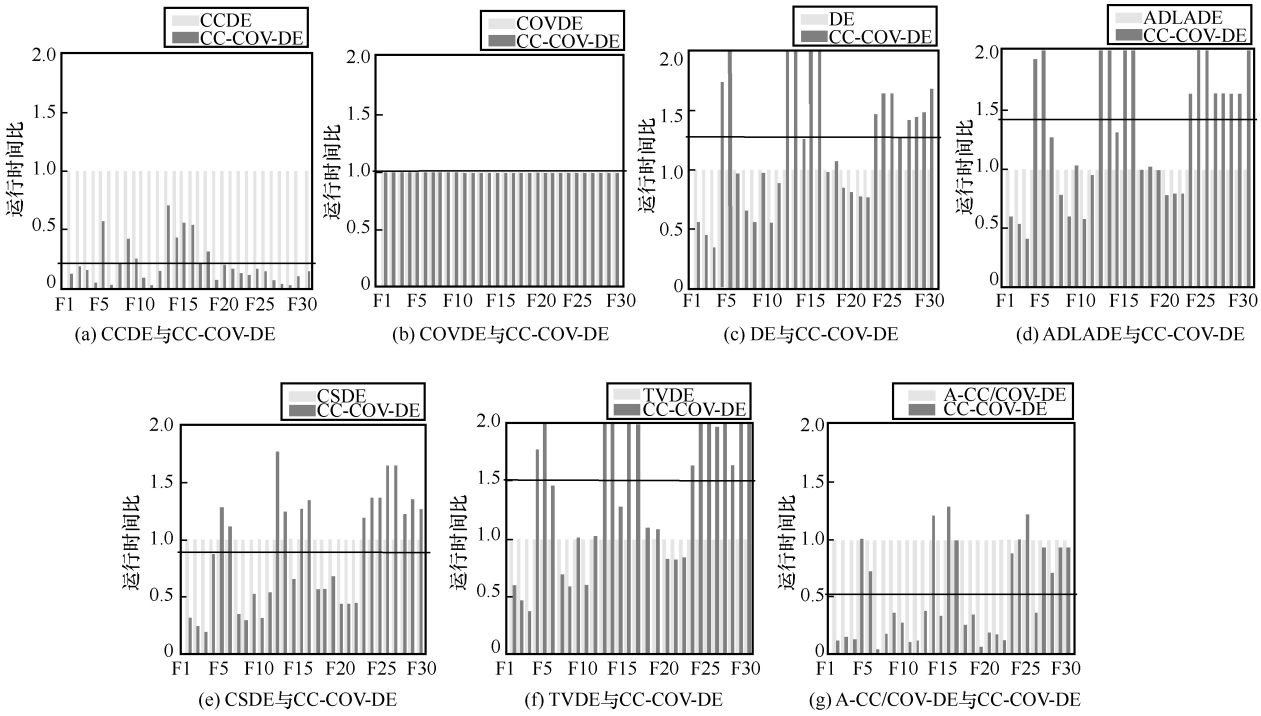

图 9　CC-COV-DE 与其他 7 个算法在 30 维的 CEC2014 测试函数集上的运行时间比

A-CC/COV-DE 的 0.211 2、1、1.271 5、1.414 6、0.888 6、1.503 0、0.516 7 倍。通过分析可得，CC-COV-DE 的计算成本高的原因是在优化子组件的过程中需要计算协方差矩阵，计算的时间成本比其他算法稍大。

4　结束语

协同进化算法在解决可分离问题时性能较好，然而随着优化问题维度的增加，子组件中有关联性的变量相互作用关系更加复杂，若不消除子组件中变量之间的相关性，则会降低种群的进化速度。

本文提出的基于协方差分析的合作协同差分进化算法使用分组策略对优化问题进行分解之后，在对子组件进行优化的过程中利用协方差分析变量的数据特征，利用子组件中的变量构造协方差矩阵，根据特征向量对原始坐标进行旋转，消除子组件中变量之间的相关性，避免进化陷入局部最优，加快种群的收敛速度。

与已有方法对比，本文使用协方差分析的方法消除子组件内部变量之间的相关性，通过提高子组件内部的进化效率，为子组件之间的协同进化提供更好的条件。协方差计算会增加一些时间开销，但总体的综合进化效率得到提升。在 CEC 2014 测试函数集上进行仿真实验，验证了所提算法的有效性。

参考文献：

[1] KESHK M, SINGH H, ABBASS H. Automatic estimation of differential evolution parameters using Hidden Markov Models[J]. Evolutionary Intelligence, 2018, 10(3/4): 77-93.

[2] CAI Y Q, WANG J H. Differential evolution with neighborhood and direction information for numerical optimization[J]. IEEE Transactions on Cybernetics, 2013, 43(6): 2202-2215.

[3] DENG L B. An adaptive dimension level adjustment framework for differential evolution[J]. Knowledge-Based Systems, 2020, 206(4): 1-20.

[4] WANG Y, CAI Z X, ZHANG Q F. Differential evolution with composite trial vector generation strategies and control parameters[J]. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 55-66.

[5] MALLIPEDDI R. Differential evolution algorithm with ensemble of parameters and mutation strategies[J]. Applied Soft Computing, 2011, 11(2): 1679-1696.

[6] SUN G J, LAN Y F, ZHAO R Q. Differential evolution with Gaussian mutation and dynamic parameter adjustment[J]. Soft Computing, 2019, 23(5): 1615-1642.

[7] SUN G J, WU Y R, DENG L B, et al. Elite representative based individual adaptive regeneration framework for differential evolution[J]. IEEE Access, 2020, 8: 61226-61245.

[8] DENG L B. ERG-DE: an elites regeneration framework for differential evolution[J]. Information Sciences, 2020, 539: 81-103.

[9] SEMNANI A, KAMYAB M, REKANOS I T. Reconstruction of one-dimensional dielectric scatterers using differential evolution and particle swarm optimization[J]. IEEE Geoscience and Remote Sensing Letters, 2009, 6(4): 671-675.

[10] FERNANDEZ-MARTINEZ J L, GARCIA-GONZALO E. Stochastic stability analysis of the linear continuous and discrete PSO models[J].

IEEE Transactions on Evolutionary Computation, 2011, 15(3): 405-423.

[11] TSENG L Y, CHEN C. Multiple trajectory search for large scale global optimization[C]//Proceedings of 2008 IEEE Congress on Evolutionary Computation. Piscataway: IEEE Press, 2008: 3052-3059.

[12] LATORRE A, MUELAS S, PEÑA J M. A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test[J]. Soft Computing, 2011, 15(11): 2187-2199.

[13] POTTER M A, JONG K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Berlin: Springer, 1994: 249-257.

[14] BERGH F V D, ENGELBRECHT A P. A Cooperative approach to particle swarm optimization[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 225-239.

[15] YANG Z Y. Large scale evolutionary optimization using cooperative coevolution[J]. Information Sciences, 2008, 178(15): 2985-2999.

[16] YANG Z Y, TANG K, YAO X. Multilevel cooperative coevolution for large scale optimization[C]//Proceedings of 2008 IEEE Congress on Evolutionary Computation. Piscataway: IEEE Press, 2008: 1663-1670.

[17] OMIDVAR M N, LI X D, MEI Y, et al. Cooperative co-evolution with differential grouping for large scale optimization[J]. IEEE Transactions on Evolutionary Computation, 2014, 18(3): 378-393.

[18] SUN Y, KIRLEY M, HALGAMUGE S K. Extended differential grouping for large scale global optimization with direct and indirect variable interactions[C]//Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. New York: ACM Press, 2015: 313-320.

[19] STORN R, PRICE K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11: 341-359.

[20] SHI Y J, TENG H F, LI Z Q. Cooperative co-evolutionary differential evolution for function optimization[C]//International Conference on Natural Computation. Berlin: Springer, 2005: 1080-1088.

[21] YANG J Y. An adaptive encoding learning for artificial bee colony algorithms[J]. Journal of Computational Science, 2019, 30: 11-27.

[22] SUN G J, YANG B, YANG Z Q, et al. An adaptive differential evolution with combined strategy for global numerical optimization[J]. Soft Computing, 2020, 24(9): 6277-6296.

[23] SUN G J, XU G N, JIANG N. A simple differential evolution with time-varying strategy for continuous optimization[J]. Soft Computing, 2020, 24(4): 2727-2747.

[24] WANG B, REN L, PRADO J D, et al. An adaptive mechanism with cooperative coevolution and covariance for differential evolution[J]. IEEE Access, 2021, 9: 99890-99904.

[作者简介]

王彬（1971– ），男，陕西西安人，西安理工大学副教授，主要研究方向为进化计算、人工智能等。

任露（1995– ），女，陕西宝鸡人，西安理工大学硕士生，主要研究方向为进化计算。

王晓帆（1976– ），男，河北石家庄人，博士，西安理工大学教授，主要研究方向为智能信息处理。

曹雅娟（1997– ），女，陕西宝鸡人，西安理工大学硕士生，主要研究方向为进化计算。