求解大规模全局优化问题的新型三层递归差分分组方法

李 飞[1,2], 刘 翔[1,2†], 徐洪斌[3], 刘建昌[4]

(1. 安徽工业大学 电气与信息工程学院, 安徽 马鞍山 243032; 2.安徽省智能破拆装备工程实验室, 安徽 马鞍山 243002;
3. 中国石油大学(华东) 海洋与空间信息学院, 山东 青岛 266580; 4. 东北大学 信息科学与工程学院, 辽宁 沈阳 110819)

摘要: 协同进化算法在求解大规模全局优化问题上具有较好的效果, 其核心思想是利用分而治之的策略将一个高维问题分解成若干个子问题, 然后分别优化每个子问题. 然而, 现有的分解方法通常需要花费大量的计算成本来获得精确的变量分组. 通过采用递归交互检测中的历史信息简化分组过程, 能够避免检测某些集合的相互关系, 本文提出了一种新型三层递归差分分组策略(NTRDG). 与其他4种现有的分组方法相比, NTRDG在不影响分组精度的情况下计算成本消耗较低. 仿真结果表明, NTRDG在求解大规模全局优化问题时具有很强的竞争力.

关键词: 全局优化; 协同进化; 分解方法; 三层递归差分分组; 递归搜索

引用格式: 李飞, 刘翔, 徐洪斌, 等. 求解大规模全局优化问题的新型三层递归差分分组方法. 控制理论与应用, 2024, 41(4): 691 – 700

DOI: 10.7641/CTA.2023.20980

New three-level recursive differential grouping method for large-scale optimization problems

LI Fei[1,2], LIU Xiang[1,2†], XU Hong-bin[3], LIU Jian-chang[4]

(1. School of Electrical and Information Engineering, Anhui University of Technology, Anhui Ma'anshan 243032, China;
2. Anhui Province Engineering Laboratory of Intelligent Demolition Equipment, Ma'lanshan 243032, China;
3. College of Oceanography and Space Informatics, China University of Petroleum (East China), Qingdao Shandong 266580, China;
4. College of Information Science and Engineering, Northeastern University, Shenyang Liaoning 110004, China)

Abstract: The cooperative coevolution algorithm performs well in solving large-scale global optimization problems. The core idea of cooperative coevolution is to utilize a divide-and-conquer strategy for decomposing high-dimensional problems into multiple subproblems, which are then processed individually and separately. However, existing decomposition methods typically require significant computational cost to obtain accurate variable grouping. To address this issue, a novel three-level recursive differential grouping strategy (NTRDG) is proposed in this paper, which simplifies the grouping process by utilizing historical information in recursive interaction detection and avoids the detection of relationships among certain sets, leading to a lower computational cost without sacrificing grouping accuracy. Simulation results demonstrate that compared to four existing methods, NTRDG exhibits a stronger competitiveness in solving large-scale global optimization problems.

Key words: global optimization; cooperative coevolution; decomposition methods; three-level recursive differential grouping; recursive search

Citation: LI Fei, LIU Xiang, XU Hongbin, et al. New three-level recursive differential grouping method for large-scale optimization problems. *Control Theory & Applications*, 2024, 41(4): 691 – 700

1 引言

在自然科学与工程技术中, 许多问题可以转化为一个最优化问题. 随着大数据、人工智能和互联网的发展, 许多优化问题往往涉及成百上千维决策变量, 此类问题被称为大规模全局优化问题 (large-scale global optimization problem, LSGOP)[1–3]. 常规进化

收稿日期: 2022−11−04; 录用日期: 2023−07−12.
†通信作者. E-mail: jeremyliu0418@163.com; Tel.: +86 13615619003.
本文责任编委: 邹云.
国家自然科学基金青年项目(61903003), 安徽省自然科学基金青年项目(2008085QE227), 安徽省高校自然科学重点研究项目(KJ2019A0051), 安徽省智能破拆装备工程实验室开放基金资助项目(APELIDE2022A007)资助.
Supported by the National Natural Science Foundation of China (61903003), the Nature Science Research Project of Anhui Province (2008085QE227), the Scientific Research Projects in Colleges and Universities of Anhui Province (KJ2019A0051) and the Open Project of Anhui Province Engineering Laboratory of Intelligent Demolition Equipment (APELIDE2022A007).

算法在求解低维单目标全局优化问题时具有较好的效果, 但是随着问题维数的增加, 问题的搜索空间会呈指数增长, 进化算法的求解效果和效率都会急剧下降, 此类问题也被称之为“维数灾难”[4]. 因此, 如何求解大规模全局优化问题是自然科学和工程技术领域的研究热点和难点.

进化算法(evolutionary algorithms, EAs)作为一类基于种群的启发式算法, 常被用于解决各种复杂问题[5–9]. 近年来, 学者们致力于大规模全局优化研究[10–11], 提出了许多性能优越的进化算法. 整体来看, 这些算法大致可分为两类: 基于非分解的进化算法[12–14]和基于分解的进化算法[15–17].

基于非分解的进化算法与传统的进化算法一样共同求解所有变量. 为了跳出局部最优解, 基于非分解的进化算法通常需要设计特殊方案来增强种群的多样性. Cheng和Jin提出了粒子群优化算法(particle swarm optimization, PSO)的两个变体: 竞争学习粒子群算法(competitive swarm optimizer, CSO)[18]和社会学习粒子群算法(social learning PSO, SL-PSO)[19]. CSO采用竞争学习机制来比较从当前种群中随机选择的两个粒子. 在竞争后, 获胜者直接进入下一代, 而输家则通过向获胜者和群体的平均位置学习来更新其位置. 在SL-PSO算法中, 首先根据所有粒子的适应度值来对其进行排序, 然后除最佳粒子外, 其余粒子则通过向更好的粒子学习来更新其位置. LaTorre等[20]提出了MOS算法, 该算法在每一代中评估所有组合算子, 并使用性能最好的算子生成后代. Hansen和Ostermeier开发的协方差矩阵自适应进化策略(covariance matrix adaptation evolution strategy, CMAES)[21]是一种先进的进化算法, 它通过自适应调整协方差矩阵和步长参数来引导种群演化, 有效地维持了种群内的遗传多样性. 此外, Molina等人[22]提出的MA-SW-Chains 算法, 通过整合多种局部搜索启发式, 为每个个体实现了差异化的局部搜索强度, 从而增强了算法的搜索能力. 然而, 随着LSGO问题的复杂化, 上述方法可能会出现优化种群过早停滞等问题, 且随着问题规模的扩大, 这些算法通常需要大量的适应度评估(fitness evaluations, FEs).

基于分解的进化算法, 也称为协同进化算法[23–24], 首先采用分解方法将大规模全局优化问题划分为若干小规模子问题, 然后使用进化算法分别求解各个子问题. 由于子问题的规模通常小于原始问题的规模, 协同进化算法可以通过分解机制节省大量的FEs. 常规的分解方法可分为动态分解方法和静态分解方法.

动态分解方法在演化过程中动态地将变量划分为不同的组. Yang等[25]提出了随机分组(random grouping, RG), 该算法在每一代中将所有变量随机分成相同大小的组. 随后, 根据历史适应度信息来确定分组大小[26]. 然而, 此类方法忽略了变量间相互作用的基本结构, 可能将存在相互关系的决策变量划分为不同的组, 从而导致算法的性能急剧下降.

静态分组方法在初始化过程时对决策变量进行分组, 演化过程中保持分组结果不变. Vanden Bergh和Engelbrecht[27]将一个D维问题分解为k个小维子问题. 然而, 这种分解方法没有考虑变量间的相互作用. Omidvar等[4]提出了一种差分分组方法(differential grouping, DG), 该方法通过检测扰动变量时的适应度变化来识别变量之间的相互作用, 然后将存在相互关系的变量放到到同一组中, 大大增加了分组准确度, 但该方法难以检测变量之间的间接相互作用. 为了缓解这一困境, Sun等[28]在其基础上提出了扩展差分分组方法(extended DG, XDG), 该方法考虑了变量间的直接和间接相互作用. Mei等[29]通过引入交互结构矩阵提出了一种全局差分分组(global DG, GDG), 该分解方法可以使用深度优先搜索或广度优先搜索来识别变量之间的相互作用. 这些分解方法分解一个D维问题时需要消耗大量的FEs来检测变量之间的相互作用. 因此, Omidvar等[30]通过估计舍入误差的大小找到一个可靠的阈值, 节省了完全可分离函数上多达一半的计算资源. Sun等[31]提出了递归差分分组方法(recursive DG, RDG), 该方法通过递归检测集合之间的相互关系, 极大地降低了FEs的消耗. 受到RDG的启发, 三层递归差分分组算法(three-level RDG, TRDG)[32]改进了RDG的分解过程. TRDG在分组阶段将一个集合分成3个子集合，分别检测每个子集合与被测集合之间是否存在相互关系, 减少了算法的递归深度.

为了进一步降低计算成本, 本文在TRDG的基础上提出了一种新型三层递归差分分组方法(new TRDG, NTRDG). 贡献如下: 1) NTRDG通过采用递归交互检测的历史信息, 判断被测集合是否只与其中一个子集合存在相互关系, 若是, 则忽略与另外两个子集合的相互关系检测; 若否, 则分别对被测集合与3个子集合进行相互关系检测. 2) TRDG在检测两个集合之间相互关系时, 需要消耗3个FEs, NTRDG通过改进交互过程, 在检测相互关系时消耗两个FEs, 能够有效降低函数适应度评估次数.

2 背景介绍

2.1 大规模优化问题

一个大规模全局优化问题可定义如下:

$$\min f(\boldsymbol{x}),\ \boldsymbol{x}=[x_1\ \ x_2\ \cdots\ x_D]\in\mathbb{R}^D, \tag{1}$$

其中: $\boldsymbol{x}$是一个D维决策向量; x_i表示第i个变量; D表示变量的数量($i\in\{1,\cdots,D\}$), 通常等于或大于

1,000; $f(\boldsymbol{x})$称为目标函数.

2.2　三层递归差分分组策略

设$X=\{x_1,\cdots,x_D\}$是决策变量的集合, U_X是决策空间$\mathbb{R}^D$中单位向量的集合. 对于任意的单位向量$\boldsymbol{u}=(u_1,\cdots,u_D)\in U_{X_1}$($U_{X_1}$是$U_X$的子集), 如果存在$x_i\notin X$, 则$u_i=0$.

如图1所示, X_1是X的子集, X_2是X中除X_1外其他决策变量组成的集合. TRDG在分组阶段首先对X_1与X_2进行交互检测, 在确定X_1与X_2存在相互关系后, 将X_2平均分成3个大小相同的子集X_3, X_4, X_5. 类似的, TRDG会递归检测所有与X_1存在相互关系的集合, 直到X_1不与任何子集交互. TRDG根据以下定理检测变量集合之间的相互关系.

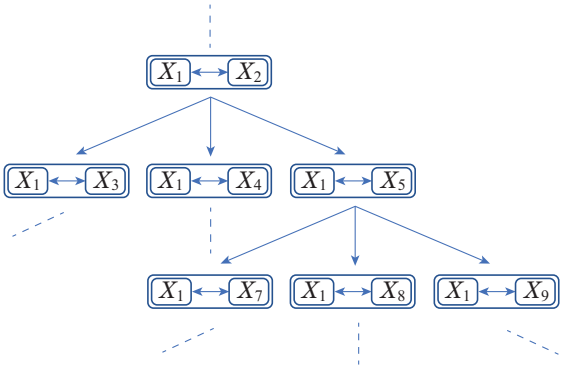

图 1　TRDG的分解过程

Fig. 1　The decomposition process of TRDG

定理 1　设f: $\mathbb{R}^D\to\mathbb{R}$是所求目标函数; $X_1\subset X$和$X_2\subset X$是函数决策变量的两个互斥子集: $X_1\cap X_2=\varnothing$. 如果在决策空间中存在两个单位向量$u_1\in U_{X_1}$和$u_2\in U_{X_2}$, 两个实数$l_1,l_2>0$, 以及一个候选解$\boldsymbol{x}^*$, 满足

$$f(\boldsymbol{x}^*+l_1u_1+l_2u_2)-f(\boldsymbol{x}^*+l_2u_2)\neq f(\boldsymbol{x}^*+l_1u_1)-f(\boldsymbol{x}^*),\tag{2}$$

则X_1和X_2中的决策变量之间存在相互作用.

由于浮点数的精度有限, 式(2)不适合用在计算设备上. 为此, 将式(2)转化为

$$f(\boldsymbol{x}^*+l_1u_1+l_2u_2)-f(\boldsymbol{x}^*+l_2u_2)-f(\boldsymbol{x}^*+l_1u_1)+f(\boldsymbol{x}^*)>\epsilon.\tag{3}$$

X_1和X_2中决策变量的相互关系对阈值ϵ非常敏感. 对于不同的问题, 识别变量间相互关系的最适阈值ϵ可能会不同. RDG采用了与GDG一样的阈值策略, 该策略基于目标值的大小来估计阈值ϵ, 但是这些阈值策略可能不适合分解不平衡问题. 受到 DG2 的启发, RDG2[31]通过导出计算舍入误差的上限来估计阈值ϵ, 该方法提高了识别变量间相互关系的准确性. TRDG采用与RDG2相同的阈值策略.

为了检测X_1和X_2之间的相互关系, TRDG需要计算Δ_1和Δ_2:

$$\Delta_1=f(\boldsymbol{x}^*+l_1u_1+l_2(u_3+u_4+u_5))-f(\boldsymbol{x}^*+l_2(u_3+u_4+u_5)),\tag{4}$$

$$\Delta_2=f(\boldsymbol{x}^*+l_1u_1)-f(\boldsymbol{x}^*).\tag{5}$$

如果式(2)不成立, TRDG可以确定X_1和X_2之间不存在交互作用. 否则, TRDG会把X_2的变量划分为3个大小相等互斥子集X_3, X_4和X_5(如图1), 并检测每个子集与X_1之间的相互作用.

为了检测X_1和X_3之间的相互关系, TRDG需要计算$\Delta_1^{'}$和$\Delta_2^{'}$:

$$\Delta_1^{'}=f(\boldsymbol{x}^*+l_1u_1+l_2u_3)-f(\boldsymbol{x}^*+l_2u_3),\tag{6}$$

$$\Delta_2^{'}=f(\boldsymbol{x}^*+l_1u_1)-f(\boldsymbol{x}^*).\tag{7}$$

为了检测X_1和X_4之间的相互关系, TRDG需要计算$\Delta_1^{''}$和$\Delta_2^{''}$:

$$\Delta_1^{''}=f(\boldsymbol{x}^*+l_1u_1+l_2u_4)-f(\boldsymbol{x}^*+l_2u_4),\tag{8}$$

$$\Delta_2^{''}=f(\boldsymbol{x}^*+l_1u_1)-f(\boldsymbol{x}^*).\tag{9}$$

为了检测X_1和X_5之间的相互关系, TRDG需要计算$\Delta_1^{'''}$ 和$\Delta_2^{'''}$:

$$\Delta_1^{'''}=f(\boldsymbol{x}^*+l_1u_1+l_2u_5)-f(\boldsymbol{x}^*+l_2u_5),\tag{10}$$

$$\Delta_2^{'''}=f(\boldsymbol{x}^*+l_1u_1)-f(\boldsymbol{x}^*).\tag{11}$$

重复上述过程, 直到TRDG寻找到与X_1有相互作用的所有变量. 在搜索的过程中, 如果X_1与子集没有相互作用, TRDG不会进一步检测X_1与这个子集的关系.

由于$X_2=X_3\cup X_4\cup X_5$, 上述各个相互关系检测过程可能存在关联.

3　新型三层递归差分分组

3.1　降低计算成本

命题 1　如果满足$(\Delta_1-\Delta_2)=(\Delta_1^{'}-\Delta_2^{'})$, 则$X_1$与$X_4$和$X_5$没有相互作用.

证　若$(\Delta_1-\Delta_2)=(\Delta_1^{'}-\Delta_2^{'})$, 又$\Delta_2=\Delta_2^{'}$, 易得$\Delta_1=\Delta_1^{'}$,

$$\Delta_1=f(\boldsymbol{x}^*+l_1u_1+l_2(u_3+u_4+u_5))-f(\boldsymbol{x}^*+l_2(u_3+u_4+u_5))=f(\boldsymbol{x}^*+l_1u_1+l_2u_3)-f(\boldsymbol{x}^*+l_2u_3),\tag{12}$$

令$\boldsymbol{x}'=\boldsymbol{x}^*+l_2u_3$, 则式(12)等价于

$$\Delta_1=f(\boldsymbol{x}'+l_1u_1+l_2(u_4+u_5))-f(\boldsymbol{x}'+l_2(u_4+u_5))=f(\boldsymbol{x}'+l_1u_1)-f(\boldsymbol{x}').\tag{13}$$

由式(13)可得X_1与X_4和X_5没有相互作用.

同理, 如果满足$(\Delta_1-\Delta_2)=(\Delta_1''-\Delta_2'')$, 则$X_1$与$X_3$和$X_5$之间没有相互关系.

由上述命题及推导可得

1) 当$(\Delta_1-\Delta_2)=(\Delta_1'-\Delta_2')$时, 不需要计算$\Delta_1''$, Δ_1'''的值就可以确定X_1与X_4和X_5不存在相互关系.

2) 当$(\Delta_1-\Delta_2)=(\Delta_1''-\Delta_2'')$时, 不需要计算$\Delta_1'''$的值就可以确定$X_1$与$X_5$之间不存在相互关系.

综上所述, 可以根据先前的相互关系检测(X_1与其中的一个或两个子集的关系), 进一步确定X_1 与其余子集之间的相互关系.

在TRDG执行的过程中, X_1与X_3, X_4, X_5之间的关系可以分为以下几种情况:

1) X_1只与X_3有相互作用, 而与X_4和X_5没有相互作用: $(\Delta_1-\Delta_2)=(\Delta_1'-\Delta_2')$;

2) X_1与X_3和X_5没有相互作用, 而只与X_4有相互作用: $(\Delta_1-\Delta_2)=(\Delta_1''-\Delta_2'')$;

3) X_1与X_3和X_4没有相互作用, 而只与X_5有相互作用: $(\Delta_1-\Delta_2)\neq(\Delta_1'-\Delta_2')\neq(\Delta_1''-\Delta_2'')$;

4) X_1与X_4和X_5都存在相互作用: $(\Delta_1-\Delta_2)\neq(\Delta_1'-\Delta_2')\neq(\Delta_1''-\Delta_2'')$;

5) X_1与X_3和X_5都存在相互作用;

6) X_1与X_3和X_4都存在相互作用;

7) X_1与X_3, X_4和X_5分别都存在相互作用; (情况5)–7)同情况4)).

对于情况1), 根据命题1可以确定X_1与X_4和X_5没有相互作用, 避免了他们之间相互关系的检测, 即避免式(8)和式(10)中Δ_1'', Δ_1'''的计算, 且X_1与X_4和X_5之间的相互关系将不作进一步研究, 这两个搜索分支将被切断. 对于情况2), 根据命题1可以确定X_1与X_3和X_5没有相互作用, 即可以避免式(10)中Δ_1'''的计算, 此搜索分支也将被切断. 对于情况3), TRDG将X_5划分为3个大小相等且相互排斥的子集X_6, X_7和X_8. 为了能够进一步检测X_1 和X_6, 检测X_7和X_8之间是否存在相互关系(如图1), 在应用命题1前, 需要知道Δ_1''' $=f(\boldsymbol{x}^*+l_1u_1+l_2u_5)-f(\boldsymbol{x}^*+l_2u_5)$的值.

在X_1与X_2的相互关系检测中, 计算Δ_1.

令$\boldsymbol{x}'=\boldsymbol{x}^*+l_2u_5$, 则

$$\Delta_1=f(\boldsymbol{x}'+l_1u_1+l_2(u_3+u_4))-f(\boldsymbol{x}'+l_2(u_3+u_4)). \tag{14}$$

如果X_1与X_3和X_4没有相互作用, 则

$$\begin{aligned}\Delta_1=&f(\boldsymbol{x}'+l_1u_1+l_2(u_3+u_4))-\\&f(\boldsymbol{x}'+l_2(u_3+u_4))=\\&f(\boldsymbol{x}'+l_1u_1)-f(\boldsymbol{x}')=\\&f(\boldsymbol{x}'+l_1u_1+l_2u_5)-f(\boldsymbol{x}^*+l_2u_5).\end{aligned} \tag{15}$$

因此, 在情况3)中, 根据命题1进一步检测X_1与X_6, X_7与X_8之间的相互关系时, 从式(15)可以看出, 可以使用等价的Δ_1来代替Δ_1''', 此时不需要计算Δ_1'''. 在第(4)至第(7)种情况中, 为了进一步检测相互关系, 仍然需要计算Δ_1'''.

对于上述的7种情况, 有3种情况(即情况1)–3))可以根据先前的相互关系的检测来确定X_1与X_4 和X_5或者是X_1与X_5之间的相互关系.

3.2 NTRDG算法描述

表1和表4分别给出了NTRDG主程序和分组过程的伪代码, 其中ub和lb分别是目标函数f决策变量的上界和下界. 在检测相互关系之前, NTRDG花费一次适应度评估求值(算法1, 步骤2). 每次算法1在调用算法4时, 都会消耗一个适应度评估来寻找变量的子分量(算法1, 步骤5). 在分组阶段, NTRDG首先检测第1个变量x_1和其余的变量(即X_1和X_2)之间的相互关系. 如果X_1和X_2之间没有相互作用, X_1中的变量将被放在一个单独的组中(算法1, 步骤9), 随后将X_2中的第一个变量移到X_1中(算法1, 步骤10). 如果X_1和X_2之间有相互作用, NTRDG将与X_1交互的变量放入X_1, 并从X_2中删除这些变量(算法1, 步骤12). 重复上述过程, 直到检测完所有变量之间的相互关系(即X_2为空).

表 1 NTRDG的主程序

Table 1 The main program of NTRDG

算法1: NTRDG的主程序
1 sets $\leftarrow\varnothing$
2 $x_{l,l}\leftarrow$ lb和$y_{l,l}\leftarrow f(x_{l,l})$
3 $X_1\leftarrow\{x_1\}$和$X_2\leftarrow\{x_2,\cdots,x_D\}$
4 while X_2不是空集合 do
5 $x_{u,l}\leftarrow x_{l,l}$, $x_{u,l}(X_1)\leftarrow$ ub(X_1), $y_{u,l}\leftarrow f(x_{u,l})$
6 $F\leftarrow\{y_{l,l},y_{u,l},\text{nan},\text{nan}\}$;
7 $(X^*,\hat{\beta})=$ Group$(X_1,X_2,x_{l,l},x_{u,l},\text{ub},\text{lb},F)$
8 if X^*与X_1相同 then
9 sets $\leftarrow$ sets$\cup X_1$
10 将X_2里的第一个变量添加到X_1中并且删除X_2里的第一个变量;
11 else
12 将X^*里的变量添加到X_1中并且从X_2中删除X_1的变量;
13 end
14 end

表2和表3分别给出了NTRDG和TRDG算法中X_1和X_2之间相互关系检测的算法伪代码. 在NTRDG中, 检测交互的算法4(如表4)需要调用算法2来判断X_1和X_2之间是否存在相互关系(算法4, 步骤1).

当X_1和X_2之间存在相互作用:

1) 如果X_2中的变量数大于2个时, X_2将被分成3个子集X_{21}, X_{22}, X_{23}(算法4, 步骤6). 根据命题2.1, 如

果X_1只与X_{21}之间存在相互作用关系(即$\beta=\beta_1$)(算法4, 步骤8), NTRDG不需要消耗冗余的适应性评估就可以确定X_1与X_{22}和X_{23}没有相互作用. 否则, 将进一步检测X_1与X_{22}之间的关系(算法4, 步骤9), 如果X_1只与X_{22}相互作用, 而与X_{21}没有相互作用(即$\beta=\beta_2$)(算法4, 步骤11), NTRDG直接可以确定X_1与X_{23}之间不存在相互关系. 如果X_1与X_{21}和X_{22}没有相互作用(算法4, 步骤12), 在算法2中的步骤2中, 因为F_3不为非数, 所以节省了用于检测X_1与X_{23}之间的相互关系的计算成本(即在算法2中的步骤3–9). 对于X_1与X_{21}, X_{22}和X_{23}的其他相互关系, 必须执行步骤15或步骤22.

表 2　NTRDG集合间相互关系的检测过程

Table 2　The procedure of detecting relationships between collections of NTRDG

算法2: $(\text{flag}, \beta)=\text{INTERACT}(X_1, X_2, x_{l,l}, x_{u,l}, \text{ub}, \text{lb}, F)$

1 将F_1, F_2, F_3, F_4置于同个集合中: $F=\{F_1, F_2, F_3, F_4\}$;
2 flag $\leftarrow$ 1;
3 if $F_3 = \text{nan}$; then
4　$x_{m,l} \leftarrow x_{l,l}$, $x_{m,l}(X_2) \leftarrow (\text{lb}(X_2)+\text{ub}(X_2))/2$
5　$x_{u,m} \leftarrow x_{u,l}$, $x_{u,m}(X_2) \leftarrow (\text{lb}(X_2)+\text{ub}(X_2))/2$;
6　$F_3 \leftarrow f(x_{m,l})$, $F_4 \leftarrow f(x_{u,m})$
7　$\Delta_1 \leftarrow (F_1-F_2)$, $\Delta_2 \leftarrow (F_1-F_2)$, $\beta \leftarrow (\Delta_1-\Delta_2)$
8　if $|\beta| \leqslant \epsilon$ then
9　　flag $\leftarrow$ 0
10　end
11 end

表 3　TRDG的交互过程

Table 3　The procedure of detecting relationships between collections of TRDG

算法3: TRDG交互检测flag = $\text{INTERACT}(X_1, X_2)$

1　$x_{u,l} = x_{l,l}$;
2　$x_{u,l} = x_{l,l}$; $x_{u,l}(X_1) \leftarrow \text{ub}(X_1)$
3　$\Delta_1 = y_{l,l} - f(x_{u,l})$ *计算适应度值变化
4　$x_{m,l} \leftarrow x_{l,l}$, $x_{m,l}(X_2) \leftarrow (\text{lb}(X_2)+\text{ub}(X_2))/2$
5　$x_{u,m} \leftarrow x_{u,l}$, $x_{u,m}(X_2) \leftarrow (\text{lb}(X_2)+\text{ub}(X_2))/2$;
6　$\Delta_2 = y_{l,l} - f(x_{u,l})$ *计算适应度值变化
7　if $|\Delta_1 - \Delta_2| > \epsilon$ then
8　　对集合X_2进行3等分并递归检测集合间的交互
9　else
10　　$X^* \leftarrow X_1$;
11　end

2) 如果X_2中只有两个变量时, 将X_2分成2个大小相等的子集X_{21}, X_{22}. 此时, 如果X_1与X_{21}有相互作用时(即$\beta=\beta_1$)(算法4, 步骤31), 则不需要消耗多余的FEs就可以确定X_1与X_{22}没有相互作用. 如果X_1与X_{21}之间不存在相互关系(算法4, 步骤32)(算法2, 步骤3), F_3判定不为非数, 所以节省了用于检测X_1与X_{22}之间相互关系的计算成本(算法2, 步骤3–9). 对于X_1与X_{21}, X_{22}都存在相互作用的情况, 必须执行步骤35. 若X_2中只有一个变量, 则不需要对X_2进行划分.

3) 重复上述过程, 直到NTRDG找到与X_1交互的所有变量. 此外, NTRDG将一个函数评估(即$f(\boldsymbol{x}_{u,l})$)从TRDG适应度检测过程(算法3, 步骤2)移至递归过程外, 在NTRDG上表现为(算法1, 步骤5), 这不会影响分解过程. 因此, 在每次交互检测的过程中, TRDG共消耗了3个适应性评估方法(算法3, 步骤2, 4, 5), 而NTRDG只消耗了两个适应度评估(算法2, 步骤6).

3.3　NTRDG交互检测实例

以一个测试函数为例, 比较NTRDG和TRDG在交互检测阶段中的差异, 这个测试函数为

$$f(x) = (x_1-x_2)^2 + (x_3-x_4)^2 + (x_5-x_6)^2 + x_7^2.$$

分组过程如图2所示. 符号↔表示集合之间存在相互关系, 而带有斜线的↔表示集合之间不存在相互关系. 在这个例子当中, $x_1 \leftrightarrow x_2$, $x_3 \leftrightarrow x_4$, $x_5 \leftrightarrow x_6$, 其中x_7为完全可分离变量.

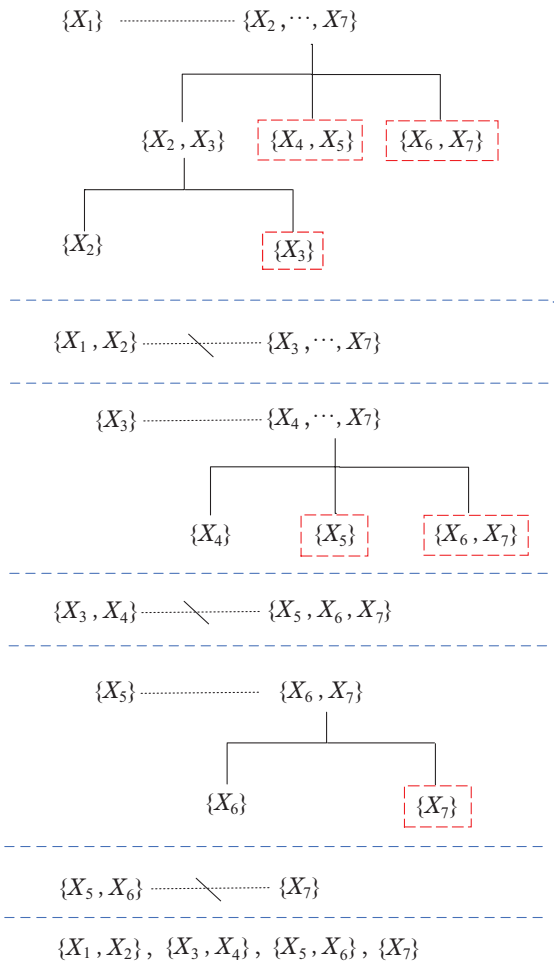

图 2　NTRDG交互检测实例

Fig. 2　The interactive detection example of NTRDG

表 4 X_2的分组策略

Table 4 The grouping strategy of X_2

算法4: $(X^*,\beta')=\text{Group}(X_1,X_2,x_{l,l},x_{u,l},\text{ub},\text{lb},F)$

(flag, β)=INTERACT($X_1,X_2,x_{l,l},x_{u,l}$, ub, lb, F)
if flag $=0$ then
  $X^*=X_1$
else
  if $|X_2|>2$ then
    将X_2平均分成3个相同大小的集合X_{21},X_{22},X_{23}
      $(X^*_{11},\hat{\beta}_1)=\text{Group}(X_1,X_{21},x_{l,l},x_{u,l},\text{ub},\text{lb}$ $\{F_1,F_2,\text{nan},\text{nan}\})$
    if $\beta\neq\hat{\beta}_1$ then
      $(X^*_{12},\hat{\beta}_2)=\text{Group}(X_1,X_{22},x_{l,l},x_{u,l},\text{ub},\text{lb}$ $\{F_1,F_2,\text{nan},\text{nan}\})$
      if $|X^*_{11}|=|X_1|$ then
        if $\beta\neq\hat{\beta}_2$ then
          if $|X^*_{12}|=|X_1|$ then
            $(X^*_{13},\beta')=\text{Group}(X_1,X_{23},x_{l,l},x_{u,l},$ ub, lb, $F)$
          else
            $(X^*_{13},\beta')=\text{Group}(X_1,X_{23},x_{l,l},x_{u,l},$ ub, lb, $\{F_1,F_2,\text{nan},\text{nan}\})$
          end
          $X^*\leftarrow X^*_{11}\cup X^*_{12}\cup X^*_{13}$
        else
          $X^*\leftarrow X^*_{11}\cup X^*_{12}$
        end
      else
        $(X^*_{13},\beta')=\text{Group}(X_1,X_{23},x_{l,l},x_{u,l},$ ub, lb, $\{F_1,F_2,\text{nan},\text{nan}\})$
        $X^*\leftarrow X^*_{11}\cup X^*_{12}\cup X^*_{13}$
      end
    else
      $X^*\leftarrow X^*_{11}$;
    end
  else if X_2中只存在两个变量 then
    将X_2平均分成两个相同大小的集合X_{21},X_{22}
      $(X^*_{11},\hat{\beta}_1)=\text{Group}(X_1,X_{21},x_{l,l},x_{u,l},$ ub, lb, $\{F_1,F_2,\text{nan},\text{nan}\})$
    if $\beta\neq\hat{\beta}_1$ then
      if $|X^*_{11}|=|X_1|$ then
        $(X^*_{12},\beta')=\text{Group}(X_1,X_{21},x_{l,l},x_{u,l},$ ub, lb, $F)$
      else
        $(X^*_{12},\beta_1)=\text{Group}(X_1,X_{21},x_{l,l},x_{u,l},$ ub, lb, $\{F_1,F_2,\text{nan},\text{nan}\})$
      end
      $X^*\leftarrow X^*_{11}\cup X^*_{12}$
    else
      $X^*\leftarrow X^*_{11}$;
    end
  else $X^*\leftarrow X_1\cup X_2$
end

NTRDG将集合$\{x_2,\cdots,x_7\}$分为3部分, 分别检测x_1与$\{x_2,x_3\}$之间的相互关系. x_1与$\{x_4,x_5\}$, $\{x_6,x_7\}$的相互关系检测过程满足式(5)的条件(x_1只与$\{x_2,x_3\}$交互), 因此省略x_1与$\{x_4,x_5\}$, $\{x_6,x_7\}$的交互检测. 继续将$\{x_2,x_3\}$集合平分(因为被测集合中只存在两个变量, 不满足3等分的条件)并检测集合x_1与x_2的交互, 得到x_1只与x_2存在相互关系, 进而省略x_1和x_3的检测过程. 输出$\{x_1,x_2\}$, 继续检测$\{x_3\}$与剩余变量的相互关系. 类似地, 在接下来的步骤中, 由于$\{x_3\}$和$\{x_4\}$, $\{x_5\}$和$\{x_6\}$分别具有相互关系唯一性, 其余的检测步骤将被省略(省略检测的集合用红色虚线标识). 最后, 将具有相互关系的变量划分为一组, 将完全可分离变量$\{x_7\}$分为一组. NTRDG处理此测试函数省略了12次相互关系检测.

TRDG在解决此问题时需要检测除x_1与x_7外的所有的被测集合以及子集合的相互关系($\{x_1\}$与$\{x_6,x_7\}$存在相互关系, 而经检测$\{x_1\}$不与$\{x_6\}$交互, 则TRDG不需要对$\{x_1\}$与$\{x_7\}$进行检测).

4 实验

4.1 测试问题和指标

选择CEC' 2010来验证算法的性能, 函数具体信息可参考文献[33]. 本文选择分组精度和FEs的平均值作为算法的性能优化评价指标, 算法独立运行25次. 分组精度的定义如下.

设$G=\{g_1,\cdots,g_m\}$表示目标函数理想的变量分组, 而$\widetilde{G}=\{\widetilde{g}_1,\cdots,\widetilde{g}_s\}$表示通过分解方法获得的变量分组. 如果$g_1$中的变量与$\widetilde{G}$中的一组变量相同, 则定义变量$s_i=1$; 否则$s_i=0$. 因此, 分组精度表示为

$$\text{Acc}=\frac{\sum\limits_{i=1}^{m}s_i}{m}. \tag{16}$$

4.2 分解性能比较

TRDG, RDG2, RDG和DG2分解方法的参数设置与原文一致. NTRDG采用了RDG2的阈值策略. 表5给出了NTRDG, TRDG, RDG2, RDG和DG2在CEC' 2010测试函数上获得的结果对比.

从表5实验结果可以看出, 除了f_6, NTRDG, TRDG, RDG2, RDG和DG2获得相同的分组精度. 在f_6上, RDG获得了理想的分组, 而NTRDG, TRDG, RDG2和DG2的分组精度很低. 这是因为它们都把大量的可分离变量放在了同一组. 注意到, 对于完全可分函数f_3, NTRDG, TRDG, RDG2, RDG和DG2得到的Acc等于0, 原因是它们将所有1000个可分离变量放入了一个不可分离组. 对于部分可分函数f_{11}, 5种方法得到的Acc都很低, 这是因为它们都把所有的500个可分离变量放在了一个不可分离组. 对于FEs这个指标,

在f_3-f_{20}上, NTRDG的性能表现要优于RDG2. 在f_1和f_2上, NTRDG和RDG2消耗的FEs是相同的, 这是由于在f_1和f_2上两个方法都不需要进行递归搜索. 与TRDG, RDG 和DG2相比, NTRDG在所有函数上消耗的FEs均明显少于它们.

总的来说, 就分组准确性而言, NTRDG, TRDG, RDG2, RDG和DG2在测试问题上表现相当. 由于阈值选取的不合适, 测试算法在CEC' 2010的某些函数上没有得到理想的变量分组. 就FEs而言, NTRDG明显优于TRDG, RDG2, RDG和DG2. 结合这两个指标, 与TRDG, RDG2, RDG和DG2对比, 可以发现NTRDG在CEC' 2010上的表现具有很强的竞争力.

表 5 NTRDG, TRDG, RDG2, RDG和DG2在CEC' 2010测试函数上的仿真结果对比

Table 5 The comparison of simulation results of NTRDG, TRDG, RDG2, RDG and DG2 on CEC' 2010 test functions

函数	NTRDG		TRDG		RDG2		RDG		DG2	
	ACC	FEs	ACC	FEs	ACC	FEs	ACC	FEs	ACC	FEs
f_1	**100%**	**2998**	**100%**	3008	**100%**	**2998**	**100%**	3008	**100%**	500501
f_2	**100%**	**2998**	**100%**	3008	**100%**	**2998**	**100%**	3008	**100%**	500501
f_3	**0%**	**3268**	**0%**	4910	**0%**	5992	**0%**	6002	**0%**	500501
f_4	**100%**	**3422**	**100%**	4109	**100%**	4198	**100%**	**4208**	**100%**	500501
f_5	**100%**	**3484**	**100%**	4067	**100%**	4144	**100%**	**4154**	**100%**	501511
f_6	0.73%	**3607**	0.11%	48848	0.11%	8563	0.11%	49880	**8.73%**	500501
f_7	**100%**	**3446**	**100%**	4088	**100%**	4222	**100%**	4232	**100%**	500501
f_8	**100%**	**4193**	**100%**	5435	**100%**	5599	**100%**	5609	**100%**	500501
f_9	**100%**	**7320**	**100%**	12791	**100%**	14026	**100%**	14036	**100%**	500501
f_{10}	**100%**	**7188**	**100%**	12665	**100%**	14008	**100%**	14018	**100%**	500501
f_{11}	**1.96%**	**6358**	**1.96%**	11744	**1.96%**	13684	**1.96%**	13694	**1.96%**	500501
f_{12}	**100%**	**7568**	**100%**	12770	**100%**	14308	**100%**	14318	**100%**	500501
f_{13}	**100%**	**13991**	**100%**	23692	**100%**	29233	**100%**	29243	**100%**	500501
f_{14}	**100%**	**9072**	**100%**	18053	**100%**	20554	**100%**	20564	**100%**	500501
f_{15}	**100%**	**9212**	**100%**	18350	**100%**	20512	**100%**	20522	**100%**	500501
f_{16}	**100%**	**9166**	**100%**	18341	**100%**	20908	**100%**	20918	**100%**	500501
f_{17}	**100%**	**9256**	**100%**	18332	**100%**	20758	**100%**	20768	**100%**	500501
f_{18}	**100%**	**23006**	**100%**	46976	**100%**	49852	**100%**	49862	**100%**	500501
f_{19}	**100%**	**3268**	**100%**	4910	**100%**	5992	**100%**	6002	**100%**	500501
f_{20}	**100%**	**26776**	**100%**	49364	**100%**	50866	**100%**	50876	**100%**	500501

4.3 NTRDG可扩展性研究

对于D维问题, DG2消耗的FEs数量是固定的: $(D^2+D+2)/2$. RDG2与RDG采用了不同的阈值策略, RDG在评估阈值时会花去少量的FEs. 综上, 本文仅研究NTRDG, TRDG和RDG2算法的可扩展性. 实验选择CEC' 2010的一些测试函数从1000维扩展到了5000 维: f_1, f_4, f_8, f_9, f_{18}, f_{19}. 对比实验结果如图3所示.

如图3(a)所示, 对于完全可分离的函数f_1, 由于所有的变量都是完全可分离的, NTRDG, TRDG和RDG2都不需要通过递归搜索来确定相互作用的变量. 因此, 3个算法的曲线几乎完全重合. 如图3(b)–(f)所示, 对于具有1个不分离组的函数f_4, f_8. 具有20个不分离组的函数f_{18}和完全不分离的函数f_{19}, 随着维数D的增加, NTRDG消耗的FEs少于TRDG和RDG2消耗的FEs, 而TRDG消耗的FEs少于RDG2消耗的FEs. 整体来看, NTRDG 与 TRDG 和 RDG2 的对比实验结果表明, NTRDG拥有更好的扩展性.

4.4 优化性能比较

在本节中, 笔者将NTRDG算法分别嵌入CMAES-CC 和 DECC 两个合作协同框架, 得到了 NTRDG–CMAES和NTRDG–DE两个算法, 并与CMAES, MOS和MA–SW–Chains 进行比较. FEs的最大数量[25]设置为3×10^6. 所有参数设置与文献[24]设置相同. 表6中分别给出所有算法在CEC' 2010测试函数上独立运行25次所获得的适应度值的平均值和标准偏差.

如表 6 所示, NTRDG–CMAES 仅在 f_5 上消耗的FEs高于CMAES. 由于NTRDG在f_3, f_{19}和f_{20}上将所有决策变量存放在同一个不可分离的组中, 两者得到的结果几乎相同. 可以发现NTRDG–CMAES比CMAES更好, 这验证了笔者的想法.

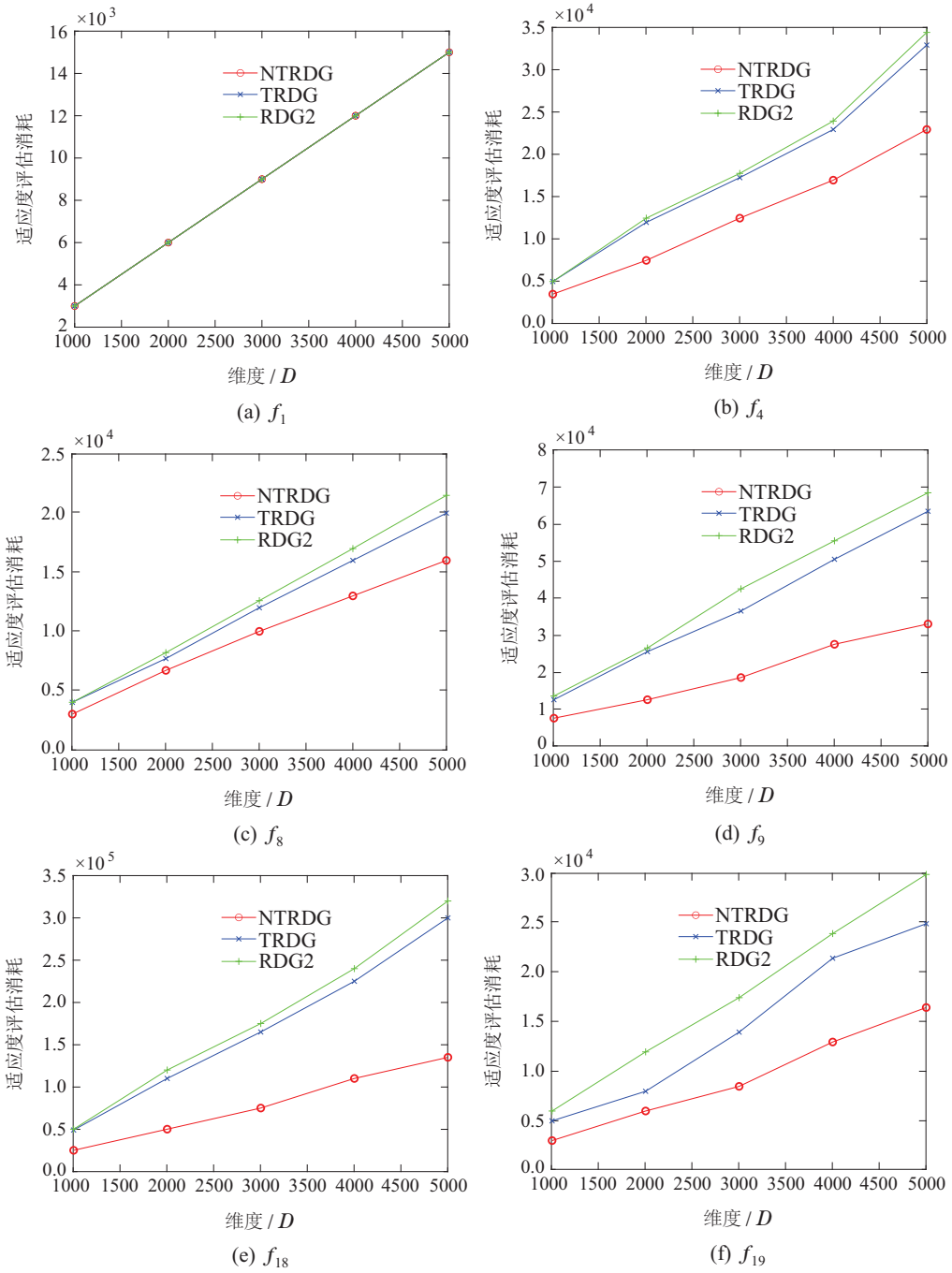

图 3 NTRDG,TRDG和RDG2在f_1, f_4, f_8, f_9, f_{18}, f_{19}上高维对比实验

Fig. 3 High-dimensional comparison experiments of NTRDG,TRDG and RDG2 on f_1, f_4, f_8, f_9, f_{18} and f_{19}

值得注意的是, NTRDG–DE将所有可分离变量放置在一个子问题中, NTRDG–CMAES则进一步将它们划分为20维的子问题. 因此, TRDG–CMAES对f_1, f_2, f_4–f_{10}, f_{12}和f_{13}使用的分解与NTRDG–DE所使用的不同, 而对于没有可分离变量的f_{14}–f_{20}, 两种算法使用的分组相同. 相较于NTRDG–DE, NTRDG–CMAES在更多函数上表现优异, 这可能表明CMAE-SCC框架使用的子优化器更有效.

可以发现在大多数测试问题上, NTRDG–CMAES明显要优于MOS和MA–SW–Chains. 发生这种情况是因为MOS和MA–SW–Chains将1000个变量一起演化, 这使得算法运行需要搜索大量空间. 因此, MOS和MA–SW–Chains可能会过早停滞. 对于f_3, f_{19}和f_{20}, NTRDG将所有1000个变量都放在同一个不可分离的集合当中. 因此, NTRDG–CMAES和NTRDG–DE在处理这些函数时也会遇到上述难题. 对于f_3, 这是因为在分解阶段选取的阈值不合适. 这意味着需要对阈值策略进一步提高. 对于f_{19}和f_{20}, 这也意味着即使对于不可分离的函数, 也需要进行适当的分解. 因此, NTRDG需要克服这些限制.

总体而言, 通过与CMAES, NTRDG–DE, MOS和MA–SW–Chains优化算法进行对比, 可以发现NTRDG–CMAES在处理CEC' 2010测试函数时表现最佳.

表 6 CMAES, MOS, MA–SW–Chains, NTRDG–CMAES和NTRDG–DE在CEC' 2010测试函数上优化仿真结果对比

Table 6 The comparison of simulation results of optimization of CMAES, MOS, MA–SW–Chains, NTRDG–CMAES and NTRDG–DE on CEC' 2010 test function

函数		CMAES	MOS	MA-SW-Chains	NTRDG-CMAES	NTRDG-DE
f_1	Mean	4.64e+06	**1.50e–28**	3.80e–14	4.89e–20	5.76e+05
f_2	Mean	5.20e+03	**0.00e+00**	8.40e+02	1.61e+03	4.36e+03
f_3	Mean	2.17e+01	**0.00e+00**	5.76e–13	2.16e+01	1.08e+01
f_4	Mean	3.84e+13	5.16e+11	2.97e+11	**9.82e+08**	3.22e+10
f_5	Mean	**6.29e+07**	4.93e+08	2.18e+08	1.21e+08	6.87e+07
f_6	Mean	1.07e+06	1.97e+07	1.42e+05	2.16e+01	**1.61e+01**
f_7	Mean	5.38e+08	3.54e+07	1.17e+02	**2.34e–18**	1.45e+04
f_8	Mean	3.43e+08	3.75e+06	6.90e+06	4.99e+07	**1.01e+06**
f_9	Mean	5.19e+06	1.13e+07	1.49e+07	**5.66e–01**	3.12e+07
f_{10}	Mean	5.16e+03	6.28e+03	2.01e+03	**1.81e+03**	3.10e+03
f_{11}	Mean	2.38e+02	3.08e+01	3.86e+01	7.23e+01	**2.59e+01**
f_{12}	Mean	3.04e–19	4.39e+03	3.24e–06	**8.27e–24**	2.97e+04
f_{13}	Mean	6.27e+04	3.32e+02	9.83e+02	**1.02e+02**	1.50e+04
f_{14}	Mean	6.07e+06	2.05e+07	3.25e+07	**6.60e–20**	2.27e+07
f_{15}	Mean	5.15e+03	1.29e+04	2.68e+03	**1.99e+03**	3.12e+03
f_{16}	Mean	4.33e+02	3.96e+02	9.95e+01	9.10e+01	**2.04e+01**
f_{17}	Mean	5.87e–11	8.45e+03	1.27e+00	**1.38e–23**	1.06e+01
f_{18}	Mean	3.71e+03	8.96e+02	1.57e+03	**1.37e+01**	1.22e+03
f_{19}	Mean	8.11e+05	5.49e+05	**3.80e+05**	1.04e+06	9.22e+05
f_{20}	Mean	8.42e+02	**9.23e+01**	1.06e+03	7.94e+02	1.15e+08

5 结论

针对三层递归差分分组算法在求解大规模全局优化问题时, 分组过程函数适应度评价次数消耗较大的难题, 本文充分采用递归搜索过程中的历史信息降低分解计算成本, 提出了一种新型三层递归差分分组策略(NTRGD). 该分组策略在CEC' 2010的仿真结果表明, NTRDG的性能明显优于其他分组策略. 此外, 将NTRDG嵌入到两个框架中去求解CEC' 2010大规模优化问题时, 所得到的算法与相关算法对比更具有竞争力.

未来研究将所提新型三层递归差分分组推广到大规模多目标优化问题[34] 和大规模双层优化问题[35], 并尝试将算法与实践相结合[36–38] .

参考文献:

[1] SONG A, CHEN W N, GU T, et al. Distributed virtual network embedding system with historical archives and set-based particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019, 51(2): 927 – 942.

[2] BENNER P. Solving large-scale control problems. *IEEE Control Systems*, 2004, 24(1): 44 – 59.

[3] SHAN S, WANG G G. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, 2009, 41(2): 219 – 241.

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

[5] GONG D, SUN J, JI X. Evolutionary algorithms with preference polyhedron for interval multi-objective optimization problems. *Information Sciences*, 2013, 233: 141 – 161.

[6] GONG D, QIN N N , SUN X Y. Evolutionary algorithms for optimization problems with uncertainties and hybrid indices. *Information Sciences*, 2011, 181: 4124 – 4138.

[7] ZHAN Z H, LI J Y, KWONG S, et al. Learning-aided evolution for optimization. *IEEE Transactions on Evolutionary Computation*, 2023, 27(6): 1794 – 1808.

[8] SUN Jing, GONG Dunwei. Recent advances in evolutionary many-objective optimization. *Control Theory & Applications*, 2018, 35(7): 928 – 938.
(孙靖, 巩敦卫. 进化高维多目标优化研究进展. 控制理论与应用, 2018, 35(7): 928 – 938.)

[9] WANG Kai, GONG Wenyin. Solving nonlinear equations system with an improved differential evolution. *Control and Decision*, 2020, 35(9): 2121 – 2128.
(王开, 龚文引. 求解非线性方程组系统的改进差分进化算法. 控制与决策, 2020, 35(9): 2121 – 2128.)

[10] MAHDAVI S, SHIRI M E, RAHNAMAYAN S. Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences*, 2015, 295: 407 – 428.

[11] LATORRE A, MUELAS S, PEA J M. A comprehensive comparison of large scale global optimizers. *Information Sciences*, 2015, 316: 517 – 549.

[12] WU Yali, FU Yulong, LI Guoting, et al. Many-objective brain storm optimization algorithm. *Control Theory & Applications*, 2020, 37(1): 193 – 204.

(吴亚丽, 付玉龙, 李国婷, 等. 高维多目标头脑风暴优化算法. 控制理论与应用, 2020, 37(1): 193 – 204.)

[13] HUANG Chenchen, WEI Xia, HUANG Deqi, et al. A shuffled frog leaping-grey wolf algorithm for solving high dimensional complex functions. *Control Theory & Applications*, 2020, 37(7): 1655 – 1666.
(黄晨晨, 魏霞, 黄德启, 等. 求解高维复杂函数的混合蛙跳–灰狼优化算法. 控制理论与应用, 2020, 37(7): 1655 – 1666.)

[14] ZHANG Guijun, WANG Xinbo, YU Li, et al. Adaptive differential evolution for high-dimension multimodal optimization problems. *Control Theory & Applications*, 2008, 25(5): 862 – 866.
(张贵军, 王信波, 俞立, 等. 求解高维多模优化问题的自适应差分进化算法. 控制理论与应用, 2008, 25(5): 862 – 866.)

[15] LIU Zhaohua, ZHANG Jing, ZHANG Yingjie, et al. Competitive-cooperative coevolutionary immune-dominant clone selection algorithm for solving the traveling salesman problem. *Control Theory & Applications*, 2010, 27(10): 1322 – 1330.
(刘朝华, 章兢, 张英杰, 等. 竞争合作型协同进化免疫算法及其在旅行商问题中的应用. 控制理论与应用, 2010, 27(10): 1322 – 1330.)

[16] WANG Ling, SHEN Jingnan, WANG Shengyao, et al. Advances in co-evolutionary algorithms. *Control and Decision*, 2015, 30(2): 193 – 202.
(王凌, 沈婧楠, 王圣尧, 等. 协同进化算法研究进展. 控制与决策, 2015, 30(2): 193 – 202.)

[17] DONG Xiaogang, DENG Changshou, TAN Yucheng, et al. Cooperative differential evolution algorithm for large-scale optimization problems. *Journal of Computer Applications*, 2017, 37(11): 3219 – 3225.
(董小刚, 邓长寿, 谭毓澄, 等. 求解大规模优化问题的新型协同差分进化算法. 计算机应用, 2017, 37(11): 3219 – 3225.)

[18] CHENG R, JIN Y. A competitive swarm optimizer for large scale optimization. *IEEE Transactions. on Cybernetics*, 2015, 45(2): 191 – 204.

[19] CHENG R, JIN Y. A social learning particle swarm optimization algorithm for scalable optimization. *Information Sciences*, 2015, 291: 43 – 60.

[20] LATORRE A, MUELAS S, PEÑA J M. A MOS-based dynamic memetic differential evolution algorithm for continuous optimization: A scalability test. *Soft Computing*, 2010, 15(11): 2187 – 2199.

[21] HANSEN N, OSTERMEIER A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 2001, 9(2): 159 – 195.

[22] MOLINA D, LOZANO M, HERRERA F. MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization. *IEEE Congress on Evolutionary Computation*. Barcelona, Spain: IEEE, 2010: 1-8.

[23] POTTER M A, JONG K A. A cooperative coevolutionary approach to function optimization. *International Conference on Parallel Problem Solving from Nature*. Berlin, Heidelberg: Springer, 1994, 866: 249 – 257.

[24] PAN Q K, GAO L, WANG L. An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems. *IEEE Transactions on Cybernetics*, 2020, 52(7): 5999 – 6012.

[25] YANG Z, TANG K, YAO X. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 2008, 178(15): 2985 – 2999.

[26] YANG Z, TANG K, YAO X. Multilevel cooperative coevolution for large scale optimization. *Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. Hong Kong: IEEE, 2008: 1663 – 1670.

[27] VANDEN BERGH F, ENGELBRECHT A P. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 2004, 8(3): 225 – 239.

[28] SUN Y, KIRLEY M, HALGAMUGE S K. Extended differential grouping for large scale global optimization with direct and indirect variable interactions. *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference-GECCO'15*. New York: ACM, 2015: 313 – 320.

[29] MEI Y, OMIDVAR M N, LI X, et al. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Transactions on Mathematical Software*, 2016, 42(2): 1 – 24.

[30] OMIDVAR M N, YANG M, MEI Y, et al. DG2: A faster and more accurate differential grouping for large-scale black-box optimization. *IEEE Transactions on Evolutionary Computation*, 2017, 21(6): 929 – 942.

[31] SUN Y, KIRLEY M, HALGAMUGE S K. A recursive decomposition method for large scale continuous optimization. *IEEE Transactions on Evolutionary Computation*, 2018, 22(5): 647 – 661.

[32] XU H B, LI F, SHEN H. A three-level recursive differential grouping method for large-scale continuous optimization. *IEEE Access*, 2020, 8: 141946 – 141957.

[33] TANG K, LI X, SUGANTHAN P N, et al. *Benchmark functions for the CEC' 2010 special session and competition on large scale global optimization*. Hefei: University of Science and Technology of China, 2010.

[34] MA X, LIU F, QI Y, et al. A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. *IEEE Transactions on Evolutionary Computation*, 2016, 20(2): 275 – 298.

[35] HUANG P Q, WANG Y. A framework for scalable bilevel optimization: Identifying and utilizing the interactions between upper-level and lower-level variables. *IEEE Transactions on Evolutionary Computation*, 2020, 24(6): 1150 – 1163.

[36] YANG Xu, WANG Rui, ZHANG Tao. Review of unmanned aerial vehicle swarm path planning based on intelligent optimization. *Control Theory & Applications*, 2020, 37(11): 2291 – 2302.
(杨旭, 王锐, 张涛. 面向无人机集群路径规划的智能优化算法综述. 控制理论与应用, 2020, 37(11): 2291 – 2302.)

[37] SUN Xiaoyan, LI Jiazhao, ZENG Bo, et al. Small-sample day-ahead power load forecasting of integrated energy system based on feature transfer learning. *Control Theory & Applications*, 2021, 38(1): 63 – 72.
(孙晓燕, 李家钊, 曾博, 等. 基于特征迁移学习的综合能源系统小样本日前电力负荷预测. 控制理论与应用, 2021, 38(1): 63 – 72.)

[38] ZHENG Xiaocao, GONG Wenyin. An improved artificial bee colony algorithm for fuzzy flexible job-shop scheduling problem. *Control Theory & Applications*, 2020, 37(6): 1284 – 1292.
(郑小操, 龚文引. 改进人工蜂群算法求解模糊柔性作业车间调度问题. 控制理论与应用, 2020, 37(6): 1284 – 1292.)

作者简介:

李 飞 副教授, 硕士生导师, 目前研究方向为进化多目标优化、鲁棒优化、昂贵大规模多目标优化和代理模型辅助进化算法, E-mail: lanceleeneu@126.com;

刘 翔 硕士研究生, 目前研究方向为大规模昂贵单目标优化, E-mail: jeremyliu0418@163.com;

徐洪斌 硕士研究生, 目前研究方向为粒子群优化、协同进化和大规模优化, E-mail: dbdxxhb@163.com;

刘建昌 教授, 博士生导师, 目前研究方向为复杂工业过程建模、控制与优化和动态多目标优化, E-mail: liujianchang@ise.neu.edu.cn.