文章编号:1671-6833(2023)05-0010-07

基于资源分配和动态分组的合作协同演化算法

欧阳聪[1], 关 静[2], 杨 鸣[1]

(1. 中国地质大学(武汉) 计算机学院,湖北 武汉 430078;2. 中国舰船研究设计中心,湖北 武汉 430064)

摘 要: 合作型协同演化算法在处理大规模全局优化问题中的决策变量完全可分或者完全不可分的问题时,精确的分组方法并不能保证提高算法性能,甚至可能会导致性能下降。针对上述问题,提出了一种基于资源分配和动态分组的合作协同演化算法(RG-CCFR3)。该算法以 CCFR3 为基础,当决策变量完全可分或完全不可分时,首先设置数组与数组索引,用于确定每轮优化时的分组大小;其次,根据分组大小对决策变量进行随机分组,使得在不同轮次的优化中每组决策变量的分配更多样化;最后,修改了 CCFR3 中每轮优化时的处理逻辑,保证了每轮优化的次数一致。通过 CEC2013 和 CEC2010 中的基准测试函数检验算法的性能,将 RG-CCFR3 与 CCFR3、MMO-CC、CBCC-RDG3 进行对比并进行显著性检验。结果表明:对比 CCFR3 算法,RG-CCFR3 算法在处理决策变量完全可分或者完全不可分的问题时,在多数情况下具有更好的性能;与 MMO-CC、CBCC-RDG3 算法相比,RG-CCFR3 算法具有一定的竞争力。

关键词:合作型协同演化; 大规模全局优化; 资源分配; 动态分组; 贡献值

中图分类号:TP301.6 **文献标志码**:A **doi**:10.13705/j.issn.1671-6833.2023.05.010

大规模全局优化(large scale global optimization, LSGO)问题是指待求解问题的维数非常高,通常涉及上千个决策变量[1]的优化问题,这类问题通常非凸、非线性并且不可微,传统的优化算法例如动态规划、牛顿法、共轭梯度法等难以解决此类问题。实际生活中优化问题随处可见,针对各种优化问题研究者提出了各种方案[2-4],特别对于 LSGO 问题,由于其复杂性,一直是各个领域亟待突破的问题。所以针对 LSGO 算法的研究是非常必要的[5-6]。

目前合作型协同演化[7](cooperative co-evolution,CC)被广泛应用于求解 LSGO 问题。合作型协同演化基于分治思想[8],将大规模全局优化问题拆分成若干个子问题分别进行求解。基于 CC 思想,目前研究者提出了基于多模态优化[9-10]的合作型协同演化、基于贡献值[11]的合作型协同演化等算法。其中基于贡献值的合作型协同演化算法被证明在求解 LSGO 问题时具有良好的性能,如 CCFR[12]、CCFR2[13]、CCFR3[14]。这类算法同样也是先将决策变量进行分组,分为若干个子问题分别进行优化。与原始合作型协同演化中选择所有亚种群进行优化不同,这类算法每次选择贡献值最大的分组进行优化。对于这类算法而言,决策变量分组的精确度会在很大程度上影响其性能,目前一些算法通过识别决策变量间的相互作用关系并将它们分配到相同的子问题中进行优化,如 DG[15]、DG2[16]、RDG[17]、RDG2[18]、ERDG[19]。

根据决策变量的分组结果,一般可将优化问题分为 3 类:决策变量完全可分、决策变量完全不可分、决策变量部分可分。通常而言,决策变量分解越精准,优化效果就会越好。然而,研究中发现在解决大规模全局优化中决策变量完全可分或完全不可分的问题时,更精准的分组有时并不会使得算法在解决该类问题时有明显的性能提升,甚至有时候会导致算法性能的下降[19]。

为了更加有效地解决此类优化问题,本文提出了一种基于资源分配和动态分组的合作协同演化算法(cooperative co-evolution algorithm based on resource allocation and dynamic grouping,RG-CCFR3)。当待优化的问题为完全可分或者完全不可分时,采用动态分组,即每轮优化时对决策变量按不同的分组大小重新分组再进行优化。实验结果表明,相比 CCFR3

收稿日期:2023-01-15;**修订日期**:2023-04-13

基金项目:国家自然科学基金资助项目(62076226)

通信作者:杨鸣(1982—),男,河北保定人,中国地质大学(武汉)副教授,博士,主要从事演化计算方面的研究,E-mail:mingyang@cug.edu.cn。

引用本文:欧阳聪,关静,杨鸣. 基于资源分配和动态分组的合作协同演化算法[J]. 郑州大学学报(工学版),2023,44(5):10-16. (OUYANG C,GUAN J,YANG M. Cooperative co-evolution algorithm based on resource allocation and dynamic grouping [J]. Journal of Zhengzhou University(Engineering Science), 2023,44(5):10-16.)

算法,该算法在处理决策变量完全可分或者完全不可分的问题时,可以达到更好的性能;与其他大规模全局优化算法相比,RG-CCFR3 也有一定的优越性。

1　CCFR3 算法的基本原理

CCFR3 算法是一种基于贡献值的合作型协同演化算法,其中贡献值体现的是每个子问题在优化后对于原始问题优化的贡献程度,贡献值越大,说明该子问题在优化后对于原始问题优化的贡献越大,反之亦然。

CCFR3 算法首先通过分组将整个问题分解为若干个子问题,然后再通过优化器算法分别求解子问题。2 个部分的算法如下所示。

算法 1　CCFR3 算法。

输入:待优化问题的决策变量;

输出:本次优化找到的最优解。

Step 1　通过分组算法将决策变量进行分组,得到分组 $C=\{C_1,C_2,\cdots,C_m\}$ 以及组数 m;

Step 2　随机生成初始规模为 N 的初始种群 P,评估所有个体并找到当前种群中的最优个体 X;

Step 3　设置贡献度数组 *deltaFit*,存放每组的贡献值;

Step 4　对每个组使用优化器算法进行优化,获得每个组的贡献值;

Step 5　每次选择贡献值最大的组进行优化,直到所有组的贡献度相等时结束本轮优化;

Step 6　若评估次数达到终止条件则程序结束,否则转到 Step 4 进行下一轮优化。

优化器算法描述的是对分解的子问题的优化,当优化满足停止条件时,将返回此次优化后该组的贡献值。

算法 2　优化器算法。

输入:种群 P、本次优化的决策变量,优化前的最优个体 X;

输出:该组的贡献值。

Step 1　根据种群和本次优化的决策变量进行优化,更新全局最优解并得到优化后种群中的最优个体 X_1;

Step 2　比较优化前的最优个体 X 和优化后种群最优个体 X_1。根据优化前后种群中最佳个体最优值的改进情况,判断是否满足提前终止条件;

Step 3　若满足提前终止条件,程序提前终止并输出本次优化该组的贡献度;

Step 4　若评估次数达到终止条件则程序结束,否则转到 Step 1。

2　RG-CCFR3 算法

RG-CCFR3 算法相比 CCFR3 算法而言,特别考虑了当分组结果为完全可分或者完全不可分的情况,增加了对这种情况的特殊处理。

当分组结果为部分可分问题时,依然采用原始 CCFR3 算法优化此类问题。当分组结果为完全可分或者完全不可分时,每轮优化时对决策变量按不同的分组大小重新分组再进行优化。相比于 CCFR3 算法,主要有以下 3 方面的变化。

(1)设置数组和索引以确定每轮优化时重新分组的分组大小。

(2)根据分组大小将所有决策变量进行重新分组。

(3)改进原始 CCFR3 算法 1 的 Step 4 和 Step 5 中每轮优化的逻辑。

后续部分将按顺序介绍 RG-CCFR3 相对 CCFR3 算法的改进。

2.1　分组大小设置

当分组结果为完全可分或者完全不可分的情况时,设置数组 *gsizeset*,$gsizeset=\{g_1,g_2,\cdots,g_n\}$,数组中的值均为整数,数组中的值代表的是重新分组的分组大小。同时还设置数组索引 *indexsize*,*indexsize* 初始值为 1。每轮优化时数组索引在数组中对应位置的值表示此次优化时的分组大小。每轮优化结束后,将 *indexsize* 的值加 1,指向数组中下一个值。当 *indexsize* 指向数组中最后一个值时,下一轮 *indexsize* 将重新设置为 1,即指向数组 *gsizeset* 第一个值。通过对 *indexsize* 的调整便可使程序在每轮优化后更换下一轮优化时确定分组大小,从而实现动态分组。图 1 展示了每轮优化时分组大小随着 *indexsize* 变化的过程。

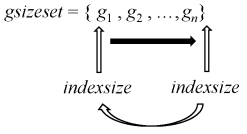


图 1　分组大小的变化

Figure 1　Changes in group size

2.2　重新分组

在上一节确定了每轮优化时的分组大小,那么现在需要按确定的分组大小将决策变量进行重新分组。

重新分组时,相对于原始 CCFR3 而言,该算法不使用算法 1 中 Step 1 通过分组算法将决策变量分解而得到的分组 $C=\{C_1,C_2,\cdots,C_m\}$,算法将对决策变量进行重新分组。算法先将待求解问题的所有决策变量按顺序排列。然后使用 MATLAB 中的 randperm 函数得到一组决策变量的随机排列,然后按 2.1 节中

确定的本轮优化的分组大小,依次将所有决策变量按分组大小进行重新分组,得到新的分组 C'。

图 2 展示了对于 1 000 个决策变量的大规模全局优化问题,将其按分组大小为 2 进行重新分组的完整过程。

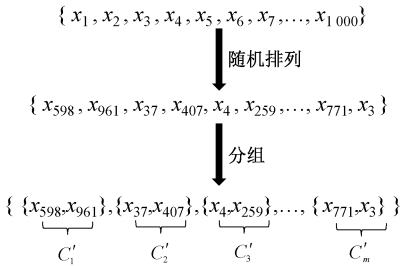


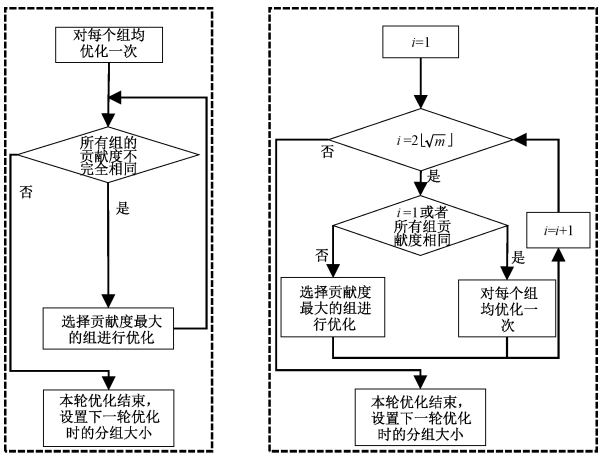
图 2 重新分组的过程

Figure 2 Regrouping process

2.3 对循环次数的改进

在原始 CCFR3 算法 1 的 Step 5 中,除非所有组的贡献度都相等,否则会不断地对贡献值最大的组进行优化。这样不方便控制每轮优化的次数,所以为保证每轮重新分组后优化的次数一致,修改了终止条件,将每一轮的优化次数设置为 $2\lfloor\sqrt{m}\rfloor$($\lfloor\ \rfloor$为向下取整符号)。在每轮优化时,若本轮第一次优化或者所有组的贡献度都相等,则对所有组优化一遍,否则对贡献度最大的组进行优化。一轮优化结束后将索引 *indexsize* 指向数组中的下一个值并继续下一轮优化,直到程序满足终止条件结束。

图 3 展示了 RG-CCFR3 和 CCFR3 每轮优化流程对比。



(a) CCFR3 (b) RG-CCFR3

图 3 CCFR3 和 RG-CCFR3 优化流程对比

Figure 3 Comparison of CCFR3 and RG-CCFR3 optimization processes

2.4 算法描述

算法 3 RG-CCFR3 算法。

输入:待优化问题的决策变量;

输出:本次优化找到的最优解。

Step 1 通过分组算法将输入的所有待优化问题的决策变量进行分组,最终得到分组 $C=\{C_1,C_2,\cdots,C_m\}$ 以及组数 m。

Step 2 根据分组 C 判断当前分组结果是否为完全可分或者完全不可分问题。如果是,则设置数组 *gsizeset* 与索引 *indexsize*,*indexsize* 初值设为 1,表示默认指向数组 *gsizeset* 中的第一个值,之后转入 Step 3;否则转入算法 1 中的 Step 2。

Step 3 随机生成初始规模为 N 的初始种群 P,评估种群 P 中的所有个体并找到当前种群中的最优个体 X。

Step 4 设置贡献度数组 *deltaFit*,用于存放每组的贡献度,每一个组的初始贡献度设置为 0。

Step 5 根据索引 *indexsize* 所指向数组 *gsizeset* 中的值确定当前分组大小,根据分组大小对决策变量进行重新分组,得到新分组 C'并更新组数 m。

Step 6 设置本轮优化次数为 $2\lfloor\sqrt{m}\rfloor$,在本轮优化中,若此次优化为本轮第一次优化或者在 *deltaFit* 数组中所有组的贡献度都相等,则对 C'中的每个组都使用优化器算法进行优化,之后更新 *deltaFit*;否则只对 *deltaFit* 数组中贡献度最大所对应的组使用优化器算法进行优化,之后更新 *deltaFit*。

Step 7 将索引 *indexsize* 加 1 后的值对数组 *gsizeset* 的长度取余,再将得到的值赋值给 *indexsize*,表示将 *indexsize* 指向下一轮优化的分组大小。

Step 8 判断当前评估次数是否达到终止条件。如果是,则输出当前的最优解并结束程序;否则转到 Step 5 进行下一轮优化。

3 实验条件

3.1 测试函数

本文使用 CEC2013 和 CEC2010 测试函数验证算法性能。

在 CEC2013 中使用 ERDG 对基准测试函数在决策变量为 1 000 维(f_{13},f_{14} 决策变量为 905 维)的情况下进行分组,用分组结果为完全可分或者完全不可分的函数来验证算法的性能,测试函数分别为 f_1、f_2、f_3、f_{12}、f_{14}、f_{15}。表 1 展示了本文实验中所使用的 CEC2013 测试函数的基本信息。

在 CEC2010 中使用 ERDG 对基准测试函数在决策变量为 2 000 维时进行分组,用其中分组结果为完全可分或者完全不可分的函数来验证算法的性能,测试函数分别是 f_1、f_2、f_3、f_{19}、f_{20}。表 2 展示了本文实验中所使用的 CEC2010 测试函数的基本信息。

表 1　CEC2013 测试函数

Table 1　CEC2013 functions

测试函数	函数名称	维度	搜索域
f_1	Elliptic Function	1 000	[-100,100]
f_2	Rastrigin Function	1 000	[-100,100]
f_3	Ackley Function	1 000	[-32,32]
f_{12}	Rosenbrock's Function	1 000	[-100,100]
f_{14}	Schwefels Function	905	[-100,100]
f_{15}	Schwefels Problem1.2	1 000	[-100,100]

表 2　CEC2010 测试函数

Table 2　CEC2010 functions

测试函数	函数名称	维度	搜索域
f_1	Shifted Elliptic Function	2 000	[-100,100]
f_2	Shifted Rastrigin's Function	2 000	[-5,5]
f_3	Shifted Ackley's Function	2 000	[-32,32]
f_{19}	Shifted Schwefel's Problem 1.2	2 000	[-100,100]
f_{20}	Shifted Rosenbrock's Function	2 000	[-100,100]

本文所使用的 CEC2013 和 CEC2010 的测试函数的全局最优解均为 0。

3.2　实验设置

种群规模 N 设置为 50,每次运行最大评估次数为 3×10^6,每个算法独立运行 25 次,每次运行使用不同的随机数种子对种群进行初始化,优化器选择 SaNSDE[20], SaNSDE 是一种基于 DE[21] 的进化算法。在重新分组的情况下,分组数组 $gsizeset=\{10,25,50,100\}$。

本文通过对比 RG-CCFR3 算法和 CCFR3 算法在测试函数独立运行 25 次的情况下测试函数解的最小值的平均值以及方差来评价算法的性能。由于平均值和方差并不能完全说明两者之间是否存在差异,所以本文通过对比 RG-CCFR3 和 CCFR3 算法在测试函数运行的实验数据进行显著性检验[22]来比较两者之间的差异。此外,本文还对比了各个算法在测试函数中的收敛曲线,收敛曲线可以反映算法能否高效地收敛至最优解。最后本文将 RG-CCFR3 算法与 MMO-CC、CBCC-RDG3 算法进行了比较。

3.3　实验环境

为了验证算法的性能,采用 MATLAB R2021b 进行编程实现该算法。实验运行环境为 64 位 Windows10 操作系统,CPU 为 Intel Xeon E5,配置 128 GB 内存。

4　实验结果以及分析

表 3 展示了 RG-CCFR3 算法和 CCFR3 算法在 CEC2013 测试函数下独立运行 25 次测试函数解的平均最小值和方差。从表 3 可知,在 6 个测试函数中,对于 f_1、f_3、f_{12}、f_{15} 函数而言, RG-CCFR3 算法的均值和方差均要比 CCFR3 小,说明收敛性和稳定性比较好,RG-CCFR3 算法性能要优于原始 CCFR3 算法。特别在 f_1、f_{12} 这 2 个测试函数上,RG-CCFR3 与 CCFR3 均值差距非常大,说明 RG-CCFR3 的性能要远优于 CCFR3。在 f_2 和 f_{14} 测试函数上,CCFR3 的性能要优于 RG-CCFR3。最后对 CCFR3 和 RG-CCFR3 算法在测试函数的实验数据集上进行了显著性检验。显著性检验结果显示,有 4 个使用 RG-CCFR3 算法的函数比 CCFR3 显著,即 f_1、f_3、f_{12}、f_{15};有 2 个使用 CCFR3 的函数比 RG-CCFR3 算法显著,即 f_2、f_{14}。

表 3　CEC2013 中每个函数独立运行 25 次的测试函数解的平均最小值和方差

Table 3　Average minimum value and variance of the test function solution for each function running independently 25 times in CEC2013

测试函数	RG-CCFR3		CCFR3	
	测试函数解的均值	测试函数解的方差	测试函数解的均值	测试函数解的方差
f_1	1.45e-19	3.36e-19	6.24e+03	1.90e+04
f_2	3.00e+01	8.10e+00	9.87e-01	1.90e+00
f_3	2.01e+01	1.01e-02	2.06e+01	1.20e-02
f_{12}	1.68e+03	1.47e+02	1.65e+09	2.82e+09
f_{14}	1.06e+11	5.58e+10	2.96e+09	2.71e+09
f_{15}	5.74e+06	3.80e+05	7.78e+06	1.70e+06

表 4 展示了 RG-CCFR3 算法和 CCFR3 算法在 CEC2010 测试函数决策变量为 2 000 的情况下独立运行 25 次测试函数解的平均最小值和方差。

表 4　CEC2010 中每个函数独立运行 25 次的测试函数解的平均最小值和方差

Table 4　Average minimum value and variance of the test function solution for each function running independently 25 times in CEC2010

测试函数	RG-CCFR3		CCFR3	
	测试函数解的均值	测试函数解的方差	测试函数解的均值	测试函数解的方差
f_1	1.95e-07	4.92e-08	4.64e+04	1.10e+05
f_2	1.58e+03	7.20e+01	2.65e-02	5.84e-03
f_3	5.58e-07	5.52e-08	1.43e+01	1.70e-01
f_{19}	6.20e+06	1.35e+06	8.27e+06	3.67e+05
f_{20}	3.93e+03	3.57e+02	1.06e+11	1.73e+10

从实验结果中可以看出,对于测试函数 f_1、f_3、f_{19}、f_{20},RG-CCFR3 算法性能要优于 CCFR3 算法,其中在 f_1、f_3、f_{20} 函数上 RG-CCFR3 算法性能相比 CCFR3 算法有显著提升。仅在函数 f_2 上,CCFR3 的性能要优于 RG-CCFR3。最后,进行了显著性检验,以比较 CCFR3 和 RG-CCFR3 算法在测试函数数据

集上的表现。显著性检验结果表明在测试函数f_1、f_3、f_{19}、f_{20}上,RG-CCFR3算法要比CCFR3算法更显著。

图4展示了RG-CCFR3算法和CCFR3算法在CEC2013测试函数独立运行25次的平均最小值随着评价次数变化的情况,即平均收敛性。通过收敛性可以反映算法能否高效地收敛至最优解,它是判断算法性能好坏与否的重要指标。

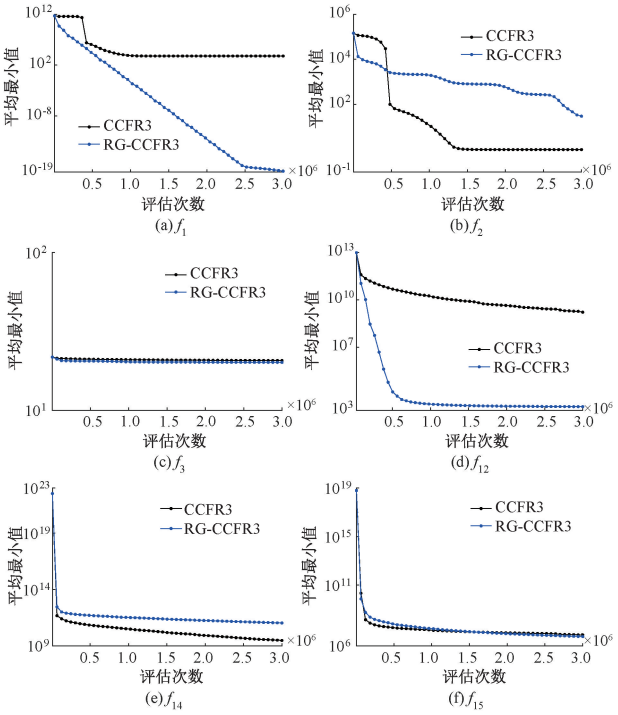


图4 CEC2013测试函数独立运行25次的平均收敛性

Figure 4 CEC2013 average convergence of 25 independent runs of each function

从图4中可以看出,对于决策变量完全可分的f_1、f_2、f_3这3个测试函数,在函数f_1和函数f_3中,RG-CCFR3收敛速度整体比CCFR3快,特别在单峰函数f_1上,RG-CCFR3收敛速度比CCFR3快,而且从图中很明显地可以看出CCFR3陷入了局部最优;在多峰函数f_2上,CCFR3整体收敛速度比RG-CCFR3快,但是CCFR3在中后半段陷入了局部最优,而RG-CCFR3从图中看仍有下降的趋势,说明RG-CCFR3相比CCFR3而言,在处理这类问题时能有效跳出局部最优;在决策变量完全不可分测试函数f_{15}上,前半段RG-CCFR3收敛速度比CCFR3慢,但在后半段体现出了RG-CCFR3的优势,在后半段RG-CCFR3收敛速度比CCFR3快,优化结束后的最小值也是RG-CCFR3更好;对于子部分重叠的测试函数f_{12}、f_{14},其中在多峰函数f_{12}上,RG-CCFR3收敛速度整体比CCFR3快很多,在单峰函数f_{14}上,CCFR3收敛速度整体比RG-CCFR3快,但相对来说差距没有特别大。这表明在子部分重叠的问题上,动态分组的策略增强了全局搜索能力,但牺牲了局部搜索能力。

图5展示了RG-CCFR3和CCFR3算法在CEC2010测试函数独立运行25次的平均收敛性。从图5可以直观看到,在决策变量完全可分的f_1、f_2、f_3这3个测试函数上,在f_1和f_3上RG-CCFR3收敛速度要明显比CCFR3快,并且CCFR3陷入了局部最优,而RG-CCFR3在这2个函数上的最小值始终呈现下降的趋势;对于函数f_2,RG-CCFR3收敛速度不如CCFR3。在决策变量完全不可分的f_{19}、f_{20}这2个测试函数上,RG-CCFR3收敛速度均要比CCFR3快。在函数f_{19}上,RG-CCFR3和CCFR3在收敛速度和最小值上比较接近;在函数f_{20}上,RG-CCFR3收敛速度明显快于CCFR3。

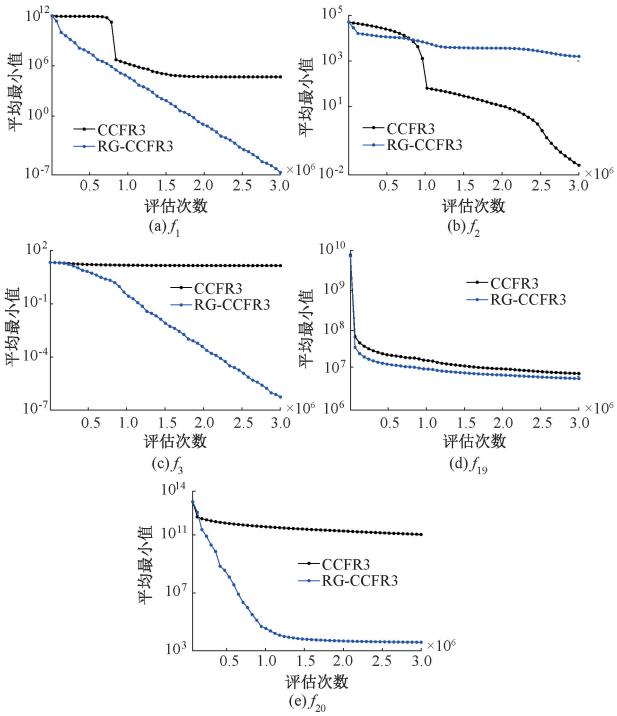


图5 CEC2010测试函数独立运行25次的平均收敛性

Figure 5 CEC2010 average convergence of 25 independent runs of each function

将RG-CCFR3与MMO-CC、CBCC-RDG3这两个解决大规模优化问题的优秀算法在CEC2013测试函数决策变量为1 000的情况下进行实验对比并进行显著性检测。MMO-CC是一种多模态优化增强的CC算法,CBCC-RDG3是CEC2019大规模全局优化竞赛的冠军算法。

表5展示了实验结果,从表5中可看出,在f_2、f_3函数上,RG-CCFR3算法性能是这3种算法中最好的。其中在函数f_2上,RG-CCFR3算法性能要明显

优于其他 2 个算法;在 f_1、f_{12}、f_{15} 函数上,RG-CCFR3 算法性能也比较接近表中对应最优算法的性能。显著性检测表明,对比 MMO-CC 算法,有 4 个使用 RG-CCFR3 算法的函数比 MMO-CC 显著,即 f_2、f_3、f_{12}、f_{15};对比 CBCC-RDG3 算法,有 3 个使用 RG-CCFR3 算法的函数比 CBCC-RDG3 显著,即 f_1、f_2、f_3。上述实验表明,对比其他大规模全局优化算法,RG-CCFR3 也有一定的竞争力。

表 5　RG-CCFR3 与其他大规模全局优化算法的在 CEC2013 上的对比测试结果

Table 5　Comparison test results of RG-CCFR3 and other large-scale global optimization algorithms at CEC2013

测试	RG-CCFR3		MMO-CC		CBCC-RDG3	
函数	测试函数解的均值	测试函数解的方差	测试函数解的均值	测试函数解的方差	测试函数解的均值	测试函数解的方差
f_1	1.45e-19	3.36e-19	4.83e-20	9.45e-21	1.17e-18	1.38e-19
f_2	3.00e+01	8.10e+00	1.53e+03	7.42e+01	2.33e+03	1.03e+02
f_3	2.01e+01	1.01e-02	2.01e+01	1.31e-02	2.04e+01	4.09e-03
f_{12}	1.68e+03	1.47e+02	8.98e+10	2.60e+11	7.12e+02	1.19e+02
f_{14}	1.06e+11	5.58e+10	3.54e+11	4.77e+11	1.38e+09	1.26e+09
f_{15}	5.74e+06	3.80e+05	4.31e+08	2.06e+08	2.23e+06	2.97e+05

5　结论

为了更好地求解大规模全局优化问题中决策变量为完全可分或者完全不可分的问题,本文提出了基于资源分配和动态分组的合作协同演化算法 RG-CCFR3。算法在 CCFR3 的基础上增加了对决策变量为完全可分或者完全不可分的问题时的处理,每轮优化时对决策变量按不同的分组大小重新分组再进行优化。本文将 RG-CCFR3 算法与原始 CCFR3 算法进行了对比实验,实验结果表明,RG-CCFR3 算法在 CEC2013 的 6 个测试函数中有 4 个函数的性能优于 CCFR3;在 CEC2010 的 5 个测试函数中有 4 个函数的性能优于 CCFR3。这表明通过重新分组,该算法在处理决策变量完全可分或者完全不可分问题时,大多数情况下会有更好的性能。本文还将 RG-CCFR3 与 MMO-CC、CBCC-RDG3 算法进行对比,结果表明,对比这些优秀的大规模全局优化算法,RG-CCFR3 也有一定的竞争力。后续的研究工作应该专注于动态分组时分组大小的确定,使得算法在处理不同完全可分或者完全不可分的大规模问题时具有更好的稳定性。

参考文献:

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

[2] LIANG J, QIAO K J, YU K J, et al. Utilizing the relationship between unconstrained and constrained Pareto fronts for constrained multiobjective optimization[J]. IEEE Transactions on Cybernetics, 2023, 53(6): 3873-3886.

[3] YU K J, ZHANG D Z, LIANG J, et al. A correlation-guided layered prediction approach for evolutionary dynamic multiobjective optimization[EB/OL]. (2022-07-22)[2022-11-13]. https://ieeexplore.ieee.org/abstract/document/9837296.

[4] LIANG J, BAN X X, YU K J, et al. A survey on evolutionary constrained multiobjective optimization[J]. IEEE Transactions on Evolutionary Computation, 2022, 27(2): 201-221.

[5] 杨振宇. 基于自然计算的实值优化算法与应用研究[D]. 合肥: 中国科学技术大学, 2010.
YANG Z Y. Research on real-valued optimization algorithm based on natural computing and its application[D]. Hefei: University of Science and Technology of China, 2010.

[6] 高岳林, 杨钦文, 王晓峰, 等. 新型群体智能优化算法综述[J]. 郑州大学学报(工学版), 2022, 43(3): 21-30.
GAO Y L, YANG Q W, WANG X F, et al. Overview of new swarm intelligent optimization algorithms[J]. Journal of Zhengzhou University (Engineering Science), 2022, 43(3): 21-30.

[7] PENG X G, LIU K, JIN Y C. A dynamic optimization approach to the design of cooperative co-evolutionary algorithms[J]. Knowledge-Based Systems, 2016, 109: 174-186.

[8] YANG P, TANG K, YAO X. A parallel divide-and-conquer-based evolutionary algorithm for large-scale optimization[J]. IEEE Access, 2019, 7: 163105-163118.

[9] PENG X G, JIN Y C, WANG H D. Multimodal optimization enhanced cooperative coevolution for large-scale optimization[J]. IEEE Transactions on Cybernetics, 2019, 49(9): 3507-3520.

[10] WU Y P, PENG X G, WANG H D, et al. Cooperative coevolutionary CMA-ES with landscape-aware grouping in noisy environments[J]. IEEE Transactions on Evolutionary Computation, 2023, 27(3): 686-700.

[11] OMIDVAR M N, LI X D, YAO X. Smart use of computational resources based on contribution for cooperative co-evo-

lutionary algorithms[C]//Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. New York: ACM, 2011: 1115-1122.

[12] YANG M, OMIDVAR M N, LI C H, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

[13] YANG M, ZHOU A M, LI C H, et al. CCFR2: a more efficient cooperative co-evolutionary framework for large-scale global optimization[J]. Information Sciences, 2020, 512: 64-79.

[14] YANG M, ZHOU A M, LU X F, et al. CCFR3: a cooperative co-evolution with efficient resource allocation for large-scale global optimization[J]. Expert Systems with Applications, 2022, 203: 117397.

[15] 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.

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

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

[18] SUN Y, LI X D, ERNST A, et al. Decomposition for large-scale optimization problems with overlapping components[C]//2019 IEEE Congress on Evolutionary Computation (CEC). Piscataway: IEEE, 2019: 326-333.

[19] YANG M, ZHOU A M, LI C H, et al. An efficient recursive differential grouping for large-scale continuous problems[J]. IEEE Transactions on Evolutionary Computation, 2021, 25(1): 159-171.

[20] YANG Z Y, TANG K, YAO X. Self-adaptive differential evolution with neighborhood search[C]//2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). Piscataway: IEEE, 2008: 1110-1116.

[21] YU K J, LIANG J, QU B Y, et al. Dynamic selection preference-assisted constrained multiobjective differential evolution[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2022, 52(5): 2954-2965.

[22] 温忠麟, 侯杰泰, 马什赫伯特. 结构方程模型检验: 拟合指数与卡方准则[J]. 心理学报, 2004, 36(2): 186-194.
WEN Z L, HOU J T, MARSH H W. Structural equation model testing: cutoff criteria for goodness of fit indices and Chi-square test[J]. Acta Psychologica Sinica, 2004, 36(2): 186-194.

Cooperative Co-evolution Algorithm Based on Resource Allocation and Dynamic Grouping

OUYANG Cong[1], GUAN Jing[2], YANG Ming[1]

(1. School of Computing, China University of Geosciences, Wuhan 430078, China; 2. China Ship Development and Design Center, Wuhan 430064, China)

Abstract: The precise grouping method might not constantly improve the algorithm performance and sometimes even lead to performance degradation when the cooperative co-evolutionary algorithm was used to solve large-scale global optimization problems with entirely separable or fully non-separable decision variables. A cooperative co-evolutionary algorithm (RG-CCFR3) based on resource allocation and dynamic grouping was proposed to address the above problems. The algorithm was based on CCFR3, where the array with the array index was first set for determining the group size at each round of optimization when the decision variables were fully divisible or fully indivisible. Secondly, the decision variables were randomly grouped according to the group size, which made the assignment of each group of decision variables more diverse in different rounds of optimization. Finally, the processing logic in CCFR3 at each round of optimization was modified to ensure a consistent number of rounds of optimization. The benchmark test functions in CEC2013 and CEC2010 were selected to examine the algorithm's performance. And RG-CCFR3 was compared with CCFR3, MMO-CC, and CBCC-RDG3 and tested for significance. The experimental results showed that, compared with the CCFR3 algorithm, the RG-CCFR3 algorithm would perform better in most cases when dealing with problems with entirely separable or non-separable decision variables; it was also competitive with the MMO-CC and CBCC-RDG3 algorithms.

Keywords: cooperative co-evolution; large scale global optimization; resource allocation; dynamic grouping; contribution