文章编号: 1001-0920 (2015) 04-0715-06 DOI: 10.13195/j.kzyjc.2014.0223

基于混合策略的双种群约束优化算法

毕晓君, 张 磊

(哈尔滨工程大学 信息与通信工程学院，哈尔滨 150001)

摘 要: 提出一种基于混合策略的双种群约束优化算法. 利用双种群存储机制处理约束条件, 并采用约束支配更新不可行解集, 同时采用混合策略进化种群: 在进化前期利用 Deb 准则产生可行解, 并保留一部分非劣不可行解参与进化, 保持种群多样性; 在进化后期让最优个体和次优个体参与进化, 使种群快速收敛. 仿真实验结果表明, 所提出的算法在保证种群多样性的同时, 能够较好地收敛到全局最优解, 且鲁棒性较好.

关键词: 约束优化；双种群；混合策略；差分进化算法

中图分类号: TP183 **文献标志码:** A

Dual population constrained optimization algorithm with hybird strategy

BI Xiao-jun, *ZHANG Lei*

(College of Information and Communication Engineering，Harbin Engineering University，Harbin 150001，China. Correspondent：ZHANG Lei，E-mail：zl12306124@163.com)

Abstract: A dual population constrained optimization algorithm with the hybrid strategy is proposed. The method uses the dual population to deal with constrains, and employs constraint control as the approach of updating infeasible sets. The hybird strategy is used to evolve population. In the early stage of the evolution, the Deb criterion is used to generate feasible solutions, and some inferior infeasible solutions are retained to participate in the evolution to ensure the diversity of the population. In the later stage of the evolution, the global optimal solution and second-best solution take part in the evolution, so as to quickly converge. The simulation results show that, while keeping the diversity of the population, the proposed algorithm can converge to the global optimal solutions with better robustness.

Keywords: constrained optimization；dual population；hybrid strategy；differential evolution algorithm

0 引 言

约束优化问题是一种广泛存在于实际应用中却较难求解的问题, 因此对其研究具有重要的实际和理论意义. 目前, 约束优化问题求解已成为进化计算领域的一个重要研究方向[1]. 约束优化问题与一般的优化问题不同, 由于其约束条件的存在, 可行域空间将变得十分的复杂. 当等式约束条件较多时, 可行域空间会十分狭小, 需要算法兼顾良好的多样性和局部搜索能力; 当存在多个不连通的可行域时, 会有多个局部最优解, 需要算法保持良好的多样性. 一般而言, 对于约束优化问题的研究主要包括两方面, 一方面是进化算法, 另一方面是约束处理技术. 近年来的研究表明, 差分进化算法是诸多进化算法中处理约束优化问题效果最好的[2-3]. 而约束处理技术中效果较好的是随机排序法、可行性准则和 ε 约束等. Runarsson 等[4]提出了随机排序法, 该方法是经典的约束处理方法, 但需要设置参数 Pf. Deb[5]提出了一种联赛准则来比较个体, 但 Deb 准则没有充分利用不可行解信息, 不利于保持种群的多样性. Takahama 等[6]提出了一种 ε 约束算法, 但该方法需要对 ε 进行微调, 并且需要结合梯度变异算子来进一步改善算法性能. 郑建国等[7]提出了一种 εDE 算法, 将差分进化算法与 ε 约束相结合, 是目前利用差分进化算法处理约束优化问题效果最好的方法, 但适应度评价次数较多. 上述的随机排序法、Deb 准则和 ε 约束都涉及可行解与不可行解之间的直接比较, 因此, 当可行解优于不可行解时, 种群

收稿日期: 2014-02-22；**修回日期:** 2014-08-26.

基金项目: 国家自然科学基金项目(61175126)；中央高校基本科研业务费专项项目(HEUCFZ1209)；教育部博士点基金项目(20112304110009)；辽宁省博士科研启动基金项目(201205118)；辽宁省教育厅科学技术研究项目(L2012458).

作者简介: 毕晓君(1964−), 女, 教授, 博士生导师, 从事信息智能处理技术、智能优化算法、数字图像处理等研究；张磊(1987−), 男, 博士生, 从事信息智能处理技术、约束多目标优化的研究.

易陷入局部最优, 当优秀的不可行解优于较差可行解时, 将不利于种群收敛. 随机排序法和 ε 约束需要通过大量的实验设置合适的且对算法性能影响较大的参数, 这样不利于在实际中的应用.

鉴于此, 本文提出一种基于混合策略的双种群约束优化算法. 该方法借鉴多目标优化中的双种群存储思想, 将可行解和不可行解分开进行存储, 并给出适用于约束单目标优化问题的双种群存储规则, 可有效避免可行解与不可行解的直接比较, 且不需要设置任何参数, 能够很好地保留优秀解. 同时, 采用约束支配更新不可行解集, 进化中采用基于混合策略的进化机制. 实验结果表明, 所提出的算法在收敛精度和收敛速度方面都取得了较好的效果.

1 约束优化问题及相关定义

不失一般性, 以最小化问题为例对约束优化问题进行如下定义:

$$\begin{aligned} &\min\ f(X).\\ &\text{s.t. } g_i(X) \leqslant 0,\ i=1,2,\cdots,p;\\ &\quad\ \ h_j(X)=0,\ j=1,2,\cdots,q. \end{aligned} \tag{1}$$

其中: $X=(x_1,x_2,\cdots,x_n)\in \boldsymbol{R}^n$ 为 n 维决策变量, $f(X)$ 为目标函数, $g_i(X)$ 为第 i 个不等式约束条件, p 为不等式约束条件个数, $h_j(X)$ 为第 j 个等式约束条件, q 为等式约束条件个数. 将等式约束条件转化为不等式约束条件[8-9]

$$|h_j(X)|-\delta \leqslant 0. \tag{2}$$

满足所有约束条件的解空间 S 称为式 (1) 的可行域, $S\subset R^n$. 包括在 S 中的个体 X 称为可行解, 否则称为不可行解.

约束违反函数如式下所示:

$$G(X)=\sum_{i=1}^{p}\max(0,g_i(X))+\sum_{j=1}^{q}\max(0,|h_j(X)|-\delta), \tag{3}$$

其大小称为约束违反度.

2 差分进化算法

目前, 大部分的约束优化算法都选取差分进化算法作为进化机制, 这是因为差分进化算法较其他进化算法在维护种群多样性和搜索能力方面效果最好[10]. 本文也采用差分进化算法, 并对其进行改进, 进一步提高约束优化性能.

差分进化算法的主要步骤包括变异、交叉、选择三种操作, 下面进行详细介绍.

Step 1: 初始化种群. 随机产生 N 个个体, 每个个体 $X_i=(x_1,x_2,\cdots,x_n)$ 的第 j 维分量为

$$x_j=l_j+\text{rand}(\ )u_j,$$
$$j=1,2,\cdots,n,\ i=1,2,\cdots,N,$$

rand( ) 为 $[0,1]$ 的随机数, $x_j\in[l_j,u_j]$.

Step 2: 进行变异操作

$$V_i(t+1)=X_{r_1}(t)+F\times(X_{r_2}(t)-X_{r_3}(t)). \tag{4}$$

其中: F 为缩放因子; r_1,r_2,r_3 为 $1,2,\cdots,N$ 上互不相等的正整数, $i=1,2,\cdots,N$, 且 r_1,r_2,r_3 与当前目标矢量索引 i 不同.

Step 3: 进行交叉操作

$$u_{ij}(t+1)=\begin{cases} v_{ij}(t+1),\ \text{rand}(j)\leqslant \text{CR};\\ x_{ij}(t+1),\ \text{otherwise}. \end{cases} \tag{5}$$

其中: t 为进化代数, v_{ij} 为个体 V_i 的第 j 维分量, x_{ij} 为个体 X_i 的第 j 维分量, CR 为交叉因子, rand(j) 为属于 $[0,1]$ 的随机数.

Step 4: 进行选择操作

$$X_i(t+1)=\begin{cases} U_i(t+1),\ f(U_i(t+1))<f(X_i(t));\\ X_i(t),\ f(U_i(t+1))\geqslant f(X_i(t)). \end{cases} \tag{6}$$

其中: t 为进化代数, $U_i=(u_{i1},u_{i2},\cdots,u_{in})$.

3 基于混合策略的双种群约束优化算法

单目标约束优化算法在进行个体之间的比较时还存在一定的缺陷, 其中较先进的基于 ε 约束的优化算法需要通过大量实验来设置对进化产生很大影响的实验参数, 并且在一些问题上存在易陷入局部最优、收敛精度不高等缺点. 本文经过深入研究, 提出一种基于混合策略的双种群约束优化算法.

3.1 基于双种群的约束处理技术改进

双种群存储机制对可行解和不可行解分别进行存储, 避免了可行解与不可行解的直接比较, 操作简单易行, 能够保持很好的多样性, 目前已应用于解决多目标约束优化问题, 并取得了较好的效果, 但在单目标约束优化问题中还较少出现相关文献. 单目标约束优化问题与多目标约束优化问题相同, 都需要在进化过程中公平比较个体, 进而选择出优秀个体进入下一代种群, 因此都可适用双种群存储思想. 目前的双种群存储技术也存在一定缺陷, 在对不可行解进行更新时, 一般都是保留约束违反度较小的不可行解, 这样的不可行解可能目标函数较差, 让其参与进化反而会影响种群质量; 同时, 双种群存储的时间复杂度相对较高, 影响算法收敛速度. 为此, 本文对双种群存储机制进行改进, 提出了更新不可行解集的新方法, 在进化前期, 保留了约束违反度和目标函数同时较优的不可行解, 提高种群质量; 当可行解集数量达到预定

的规模时, 将其作为进化种群而不是归档集, 加快种群收敛速度; 在进化后期, 不再更新和利用不可行解集, 从而加快收敛速度. 在本文提出的改进双种群存储机制中, 给出两个更新不可行解集的新定义.

定义 1 (约束支配)　不可行解 X 的约束向量 $(f(X), G(X))$ 中的元素均优于不可行解 Y 的约束向量 $(f(Y), G(Y))$ 中的元素, 称不可行解 X 约束支配不可行解 Y.

定义 2 (非劣不可行解)　不可行解集中不存在 X' 约束支配 X^*, 称 X^* 为非劣不可行解.

根据定义 1 和定义 2, 本文提出的双种群存储方式如下: 首先更新不可行解集, 将上一代种群中的不可行解集 $\mathrm{Pcc}(t-1)$ 与当代不可行解集 $\mathrm{Pcc}'(t)$ 合并, 根据定义 1 和定义 2 从中选取最优的 N_2 (不可行解集的大小) 个个体作为下一代不可行解种群 $\mathrm{Pcc}''(t+1)$. 如果最优个体的数量大于 N_2, 则再选取约束违反度最小的 N_2 个个体. 然后更新可行解集, 在进化初期, 可行解的个数可能较少, 为了快速产生 N_1 个 (可行解集的大小) 可行解, 利用 Deb 准则比较个体, 同时将上一代种群中的可行解集 $\mathrm{Pff}(t-1)$ 与当代可行解集 $\mathrm{Pff}'(t)$ 合并, 从中选取目标函数值最小的 N_1 个个体作为下一代可行解种群 $\mathrm{Pff}''(t+1)$. 当可行解集个数达到 N_1 时, 停止使用 Deb 准则, 将可行解集作为进化种群, 同时将上一代种群中的可行解集与当代可行解集合并, 从中选取目标函数值最小的 N_1 个个体作为下一代种群.

本文采用的双种群存储机制可以通过约束支配更新不可行解集, 保留约束违反度和目标函数值同时较优的个体. 首先, 通过保留优秀不可行解扩大搜索范围, 增强算法的探索能力, 提高种群多样性, 避免陷入局部最优. 其次, 避免了种群中存在约束违反度小但目标函数值较差的个体, 这样的个体往往会由于距离全局最优解较远, 从而影响种群的收敛性, 所以保留目标函数较优的不可行解有利于提高算法的搜索效率, 加快种群的收敛速度. 同时, 进化后期不再更新和利用不可行解集, 从而加快收敛速度. 另一方面, 在可行解集个数达到 N_1 时, 将可行解集作为进化种群, 而不是像文献 [11-12] 中将其作为外部存档集, 这样也会加快种群的收敛速度.

3.2　差分进化算法的改进

基于差分进化算法的约束优化算法在处理决策变量维数较高、等式约束条件较多的问题时, 仍存在易陷入局部最优、收敛精度不高等缺点. 主要原因是对搜索区域的探索能力不够, 以致种群容易陷入局部最优. 因此, 本文对差分进化算法的变异操作和初始种群的选取方法进行了改进.

3.2.1　变异操作的改进

差分进化算法中的变异操作对进化过程具有重要的作用, 但在约束优化问题中, 变异操作对搜索区域探索能力不强, 种群易陷入局部最优, 所以, 本文在进化前期让一部分优秀不可行解参与进化. 一方面, 当可行域边界离全局最优解很近时, 让优秀不可行解 (约束违反较小即离可行域边界近、目标函数较优) 参与进化, 这样的不可行解一般离全局最优解较近, 有利于种群向全局最优解靠近; 另一方面, 当存在多个孤立的可行域或可行解域较小时, 求解的问题一般会存在局部最优解, 让不可行解参与进化, 可以加大搜索空间的范围, 提高种群的多样性, 避免陷入局部最优, 从而有利于搜索到更优可行解, 继而向全局最优方向靠近. 因此, 前期变异操作为

$$\begin{aligned} &V_i(t+1) = \\ &X_{r_1}(t) + F \times (X_{\mathrm{infea}}(t) - X_{r_2}(t)). \end{aligned} \tag{7}$$

其中: F 为缩放因子, $X_{\mathrm{infea}}(t)$ 为 t 代中随机选择不可行解集中的不可行解, $X_{r_1}(t)$、$X_{r_2}(t)$ 为 t 代中随机选择可行解集中的可行解, r_1, r_2 为 $1, 2, \cdots, N_1$ 上互不相等的正整数.

在进化后期, 为了使种群更快收敛, 让上一代最优可行解 (目标函数值最小的可行解) 和次优可行解 (目标函数值次小的可行解) 同时参与进化. 为此, 后期变异操作为

$$\begin{aligned} &V_i(t+1) = \\ &X_{\mathrm{best}}(t) + F \times (X_{\mathrm{second}}(t) - X_{r_1}(t) + \\ &F \times (X_{r_2}(t) - X_{r_3}(t)). \end{aligned} \tag{8}$$

其中: F 为缩放因子, $X_{\mathrm{best}}(t)$ 和 $X_{\mathrm{second}}(t)$ 分别为 t 代中最优可行解和次优可行解, $X_{r_1}(t)$、$X_{r_2}(t)$、$X_{r_3}(t)$ 为随机选择的可行解, r_1, r_2, r_3 为 $1, 2, \cdots, N_1$ 上互不相等的正整数.

关于前期后期的划分可根据实际问题确定, 这里以最大迭代次数的一半为划分点.

3.2.2　初始种群的选取

一般在求解约束优化问题之前, 全局最优解的位置无从得知, 如果随机初始化种群不具代表性, 则不利于搜索到全局最优解, 导致达到最优解时迭代次数的增多, 从而增加算法计算量. 所以本文采用文献 [13] 中的佳点集方法, 产生均匀分布的初始化种群保持初始种群的多样性.

佳点集的构造方法如下: 点集 (n 个点)

$$P_n(i) = \{\{{r_1}^{(n)} \times i\}, \{{r_2}^{(n)} \times i\}, \cdots, \{{r_s}^{(n)} \times i\}\},$$
$$i = 1, 2, \cdots, n,$$

其偏差为

$$\varphi(n)=C(r,\varepsilon)n^{(-1+\varepsilon)}.$$

其中 $C(r,\varepsilon)$ 为只与 $r,\varepsilon(\varepsilon>0)$ 有关的常数, 则称 $P_n(i)$ 为佳点集. 取

$$r_k=\left\{2\cos\frac{2\pi k}{p}\right\},\ 1\leqslant k\leqslant s,$$

p 为满足 $(p-3)/2\geqslant s$ 的最小素数, 也可以取 $r_k=\{e^k\}, 1\leqslant k\leqslant s$. $\{r_k\times i\}$ 表示取 1 的模.

3.3 本文算法的基本流程

基于混合策略的双种群约束优化算法的主要流程如下.

Step 1: 设置所需参数, 包括初始种群规模 N、可行解集规模 N_1、不可行解集规模 N_2、缩放因子 F、交叉概率因子 CR、最大迭代次数 $G_{\max}$. 利用佳点集初始化种群.

Step 2: 计算每个个体的目标函数值, 根据式 (3) 计算每个个体的约束违反度, 更新初始双种群.

Step 3: 如果可行解集规模不为 N_1, 即小于 N_1, 则按式 (4)、(5) 进化种群, 计算个体的目标函数值和约束违反度, 采用 Deb 准则并利用第 2.2 节方法更新双种群, 否则转至 Step 4.

Step 4: 如果 $t<0.5\times G\max$, 随机生成 [0,1] 之间的随机数 $\mathrm{rand}(\ )<0.75$, 则按式 (4)、(5) 进化种群, 计算个体的目标函数值和约束违反度, 并利用第 2.2 节方法更新双种群; 如果 $\mathrm{rand}(\ )\geqslant 0.75$, 则按式 (5)、(7) 进化种群, 计算个体的目标函数值和约束违反度, 并利用第 2.2 节方法更新双种群, 否则转至 Step 5.

Step 5: 如果 $0.5\times G_{\max}\leqslant t<G_{\max}$, 则按式 (5)、(8) 进化种群, 计算个体的目标函数值和约束违反度, 并利用第 2.2 节方法更新可行解集, 否则转至 Step 6.

Step 6: 当 $t=G_{\max}$ 时算法结束, 将可行解集中的最优解作为结果输出.

算法流程如图 1 所示.

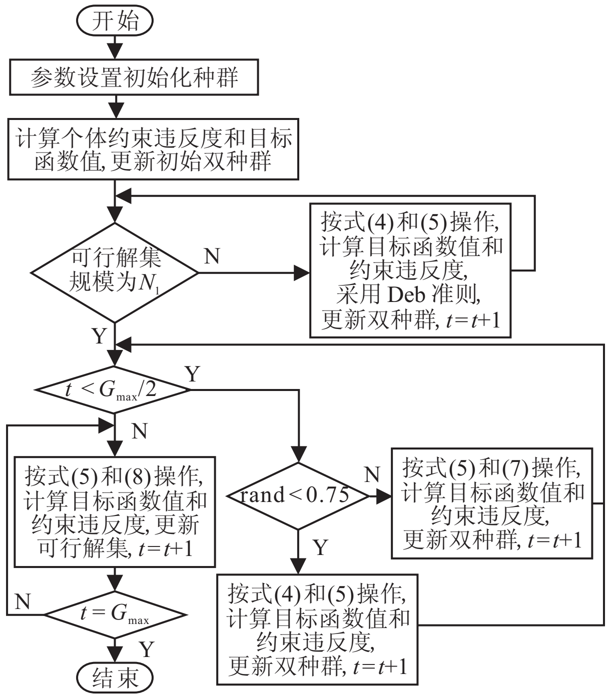

图 1 算法流程

表 1 13 个测试函数的实验参数设置

函数	种群大小	可行解集大小	不可行解集大小	最大迭代次数	缩放因子	交叉因子
g01	100	100	10	2 000	0.7	0.8
g02	200	200	10	8 000	0.5	0.5
g03	150	150	10	8 000	0.35	0.6
g04	100	100	10	2 000	0.7	0.8
g05	100	100	10	6 000	0.7	0.8
g06	100	100	10	500	0.7	0.8
g07	100	100	10	4 000	0.7	0.8
g08	100	100	10	2 000	0.7	0.8
g09	100	100	10	2 000	0.7	0.8
g10	100	100	10	5 000	0.7	0.8
g11	100	100	10	500	0.7	0.8
g12	100	100	10	500	0.7	0.8
g13	100	100	10	6 000	0.8	0.9

4 实验分析

本文所有实验在硬件配置为 Intel Pentium、G 620 CPU、4 G 内存、主频 2.6 G Hz 的计算机上进行, 程序采用 Matlab R 2010 编写.

为了验证本文算法在约束优化问题上的求解性能, 采用 13 个国际通用测试函数 (g01 ∼ g13)[13] 进行测试, 并与目前较为先进的 ISR 算法[14]、ε DE 算法[7]、SMES 算法[15] 和 ε RED 算法[16] 进行对比实验. 算法实际参数设置如表 1 所示, 所有实验参数的取值均通过反复实验验证或经验得到. 其中所有测试函数均取值 $\delta=0.000\,1$, 为了保证真实性, 在相同参数设置下分别独立运行 50 次, 所得实验结果如表 2 所示.

由表 2 可见, 本文算法对 11 个测试函数 (g02、g13 除外) 进行 50 次独立运行时都一致达到了最优解, 其中 g02 为 45 次, g13 为 45 次; ISR 只在 9 个测试函数 (g02、g07、g10、g13 除外) 上一致达到最优解; ε DE 在 11 个测试函数 (g02、g13 除外) 上一致达到最优解; SMES 只在 7 个测试函数 (g02、g05、g06、g07、g09、g13 除外) 上一致达到最优解; ε RDE 在 11 个测试函数 (g02、g07 除外) 上一致达到最优解, 其独立实验次数为 30 次. 本文算法基本上在所有的测试函数上都优于 ISR 和 SMES, 且相较于本文算法, SMES 在 g05、g06、g07、g09 上, ISR 在 g07、g09 上都未完全一致收敛, 表明由于多样性维护方面存在的缺陷导致算法最终没有收敛到全局最优解. 由此可见: 本文算法在提高种群的多样性时具有较好性能. 在函数 g02、g13 上本文算法的平均值、标准差略劣于 ε DE,

表 2　本文算法与其他 4 种算法的比较结果

函数/理论最优解	指标	算 法				
		ISR	εDE	SMES	εRDE	本文算法
g01/−15.000	最优解	−15.000	−15.000	−15.000	−15.000 000	−15.000 000
	均值	−15.000	−15.000	−15.000	−15.000 000	−15.000 000
	最差解	−15.000	−15.000	−15.000	−15.000 000	−15.000 000
	标准差	1.3 e-13	0	0	0	0
g02/−15.000	最优解	−0.803 619	−0.803 619	−0.803 601	−0.803 618	−0.803 619
	均值	−0.772 078	−0.803 004	−0.785 238	−0.803 614	−0.802 518
	最差解	−0.683 055	−0.792 608	−0.751 322	−0.803 605	−0.792 608
	标准差	2.6e-02	2.5e-03	1.67e-02	3.027e-06	3.337e-03
g03/−15.000	最优解	−1.001	−1.000	−1.000	−1.000 500	−1.000 500
	均值	−1.001	−1.000	−1.000	−1.000 500	−1.000 500
	最差解	−1.001	−1.000	−1.000	−1.000 498	−1.000 500
	标准差	6.0e-09	3.9e-06	2.09e-04	4.372e-07	3.540e-09
g04/−15.000	最优解	−30 665.539	−30 665.539	−30 665.539	−30 665.538 672	−30 665.538 672
	均值	−30 665.539	−30 665.539	−30 665.539	−30 665.538 672	−30 665.538 672
	最差解	−30 665.539	−30 665.539	−30 665.539	−30 665.538 672	−30 665.538 672
	标准差	2.2e-11	2.1e-05	0	0	0
g05/−15.000	最优解	5 126.498 1	5 126.498	5 126.599	5 126.496 714	5 126.496 714
	均值	5 126.498 1	5 126.498	5 174.492	5 126.496 714	5 126.496 714
	最差解	5 126.498 1	5 126.498	5 304.167	5 126.496 714	5 126.496 716
	标准差	6.2e-12	1.7e-05	5.006e+01	0	2.492e-06
g06/−15.000	最优解	−6 961.814	−6 961.814	−6 961.814	−6 961.813 876	−6 961.813 876
	均值	−6 961.814	−6 961.814	−6 961.284	−6 961.813 876	−6 961.813 876
	最差解	−6 961.814	−6 961.814	−6 952.482	−6 961.813 876	−6 961.813 876
	标准差	6.4e-12	2.3e-08	1.85e+00	2.803e−12	1.837e−12
g07/−15.000	最优解	24.306	24.306	24.327	24.306 209	24.306 209
	均值	24.306	24.306	24.475	24.306 210	24.306 209
	最差解	24.308	24.306	24.843	24.306 215	24.306 209
	标准差	2.7e-04	6.3e-06	1.32e-01	1.406e-06	1.063e-11
g08/−15.000	最优解	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825
	均值	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825
	最差解	−0.095 825	−0.095 825	−0.095 825	−0.095 825	−0.095 825
	标准差	4.2e-17	8.4e-17	0	0	0
g09/−15.000	最优解	680.630	680.630	680.632	680.630 057	680.630 057
	均值	680.630	680.630	680.643	680.630 057	680.630 057
	最差解	680.630	680.630	680.719	680.630 057	680.630 057
	标准差	4.6e-13	2.2e-07	1.55e-02	0	0
g10/−15.000	最优解	7 049.248	7 049.248	7 051.903	7 049.248 021	7 049.248 021
	均值	7 049.249	7 049.248	7 253.047	7 049.248 021	7 049.248 021
	最差解	7 049.296	7 049.248	7 638.366	7 049.248 022	7 049.248 021
	标准差	4.9e-03	9.0e-06	1.36e+02	2.125e-07	4.145e-12
g11/−15.000	最优解	0.750	0.75	0.750	0.749 900	0.749 900
	均值	0.750	0.75	0.750	0.749 900	0.749 900
	最差解	0.750	0.75	0.750	0.749 900	0.749 900
	标准差	1.8e-15	6.9e-14	1.52e-04	0	0
g12/−15.000	最优解	−1.000	−1.000	−1.000	−1.000 000	−1.000 000
	均值	−1.000	−1.000	−1.000	−1.000 000	−1.000 000
	最差解	−1.000	−1.000	−1.000	−1.000 000	−1.000 000
	标准差	9.6e-10	0	0	0	0
g13/−15.000	最优解	0.053 942	0.053 949	0.053 986	0.053 942	0.053 942
	均值	0.096 276	0.069 631	0.166 385	0.053 942	0.092 428
	最差解	0.438 803	0.438 846	0.468 294	0.053 942	0.438 803
	标准差	1.2e-01	7.2e-02	1.77e-01	0	1.166e-01

但在 g02 上, 本文算法最差值略优, 在其他函数上本文算法的标准差几乎均优于 εDE, 表明本文算法整体的鲁棒性更好; 在函数 g02、g13 上本文算法的最差解和标准差劣于 εRDE, 但是其独立运行次数只有 30 次, 在函数 g03、g07、g10 上, 本文算法的最差解和标准差都更优. 同时可以看出, 本文算法的标准差普遍较小, 表明本文算法较 4 种优秀算法的鲁棒性更好. 综上, 本文算法在处理 13 个基本测试函数时都具有

一定的优势.

函数g02的可行域空间较大, 决策变量的维数(20维) 也较多, 5种算法均未完全一致达到最优值, 这需要在进化中更好地保持多样性, 避免陷入局部最优. 函数g13的等式约束条件较多, 从而可行域空间较小, 大多数算法未完全达到一致最优值, 需要进一步加强种群的探索能力和开发能力, 维持多样性和收敛性. 可以看出, 对于决策变量维数较多、等式约束条件较多的问题, 大多数算法都存在一定的缺陷, 这需要研究者进行更深入的研究.

5 结 论

针对约束优化问题可行域复杂、多变等特点, 本文提出基于混合策略的双种群约束优化算法. 仿真实验结果表明, 所提出的基于混合策略的双种群约束优化算法在保证种群多样性的同时, 能够较快收敛到全局最优解, 并且鲁棒性较好. 从整体性能看, 所提出算法具有一定的优势, 但在处理决策变量维数较高、可行域空间很小的测试函数时, 与其他算法一样, 还存在一定的缺陷, 仍需作进一步深入的研究.

参考文献(References)

[1] 王勇, 蔡自兴, 周育人. 约束优化进化算法[J]. 软件学报, 2009, 20(1): 11-29.
(Wang Y, Cai Z X, Zhou Y R. Constrained optimization evolutionary algorithm[J]. J of Software, 2009, 20(1): 11-29.)

[2] Tinoco J C V, Coello C A C. hypDE: A hyper-heuristic based on differential evolution for solving constrained optimization problems[M]. Berlin: Springer, 2013: 267-282.

[3] Mezura-Montes E, Coello Coello C A. Constraint handling in nature-inspired numerical optim- ization: past, present and future[J]. Swarm and Evolutionary Computation, 2011, 1(4): 173-194.

[4] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization[J]. IEEE Trans on Evolutionary Computation, 2000, 4(3): 284-294.

[5] Deb K. An efficient constraint handling method for genetic algorithms[J]. Computer Methods in Applied Mechanics and Engineering, 2000, 186(2): 311-338.

[6] Takahama T, Sakai S. Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites[C]. Proc of the 2006 IEEE Congress on Evolutionary Computation. Singapore: IEEE Press, 2006: 1-8.

[7] 郑建国, 王翔, 刘荣辉. 求解约束优化问题的DE算法[J]. 软件学报, 2012, 23(9): 2374-2387.
(Zheng J G, Wang X, Liu R H. ε-differential evolution algorithm for constrained optimization problems[J]. J of Software, 2012, 23(9): 2374-2387.)

[8] Wang Y, Cai Z. Combining multi-objective optimization with differential evolution to solve constrained optimization problems[J]. IEEE Trans on Evolutionary Computation, 2012, 16 (1): 117-134.

[9] Liang J J, Runarsson T P, Mezura-Montes E. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization[R]. Singapore: Nanyang Technological University, 2006.

[10] Das S, Suganthan P N. Differential evolution: A survey of the state-of-the-art[J]. IEEE Trans on Evolutionary Computation, 2011, 15(1): 4-31.

[11] 孟红云, 张小华, 刘三阳. 用于约束多目标优化问题的双群体差分进化算法[J]. 计算机学报, 2008, 31(2): 228-235.
(Meng H Y, Zhang X H, Liu S Y. A differential evolution based on double population for constrained multi-objective optimization problem[J]. Chinese J of Computers, 2008, 31(2): 228-235.)

[12] 毕晓君, 王艳娇. 约束多目标人工蜂群算法[J]. 吉林大学学报, 2013(2): 397-403.
(Bi X J, Wang Y J. Constraint multi-objective evolutionary algorithm based on artificial bee colony algorithm[J]. J of Jilin University, 2013(2): 397-403.)

[13] 肖赤心, 蔡自兴, 王勇, 等. 一种基于佳点集原理的约束优化进化算法[J]. 控制与决策, 2009, 24(2): 249-253.
(Xiao C X, Cai Z X, Wang Y, et al. Constrained optimization evolutionary algorithm based on good lattice points principle[J]. Control and Dicision, 2009, 24(2): 249-253.)

[14] Runarsson T P, Yao X. Search biases inconstrained evolutionary optimization[J]. IEEE Trans on Systems Man and Cybernetics, Part C: Applications and Reviews, 2005, 35(2): 233-243.

[15] Mezura-Montes E, Velázquez-Reyes J, Coello Coello. A promising infeasibility and multiple offspring incorporated to differential evolution for constrained optimization[C]. Proc of the 2005 Conf on Genetic and Evolutionary Computation. New York: ACM, 2005: 225-332.

[16] Tetsuyuki Takahama, Setsuko Sakai. Efficient constrained optimization by the constrained rank-based differential evolution[C]. Proc of the 2012 IEEE Congress on Evolutionary Computation. Brisbane: IEEE Press, 2012: 10-15.

(责任编辑: 郑晓蕾)