

引用格式: 栗朝阳, 封全喜, 韦彦婷, 等. 改进的自适应约束差分进化算法[J]. 微电子学与计算机, 2019, 36(9): 30-37. [SU Z Y, FENG Q X, WEI Y T, et al. Improved self-adaptive constrained differential evolution algorithm[J]. Microelectronics & Computer, 2019, 36(9): 30-37.]

改进的自适应约束差分进化算法

栗朝阳, 封全喜, 韦彦婷, 苑赛楠

(1 桂林理工大学 大数据处理与算法技术研究中心, 广西 桂林 541006;

2 桂林理工大学 理学院, 广西 桂林 541006)

摘 要: 为了更好地求解约束优化问题, 提出了一种新的差分进化算法. 首先该算法利用两个变异策略平衡局部搜索与全局搜索. 其次用可行性规则选出新种群, 并利用替换策略替换新种群中的较差个体. 然后使用一种劣解变异机制, 对最差个体进行变异以保持种群多样性. 最后引入自适应参数控制机制, 增强算法的鲁棒性和适应性. 为了验证该算法的有效性, 算法测试了 10 个标准约束优化问题和 10 个工程约束优化问题, 实验结果表明, 该算法求解精度较高、具有较快的收敛速度和较强的鲁棒性.

关键词: 差分进化算法; 约束优化; 参数自适应; 可行性规则

中图分类号: TP18

文献标识码: A

文章编号: 1000-7180(2019)09-0030-08

Improved self-adaptive constrained differential evolution algorithm

SU Zhao-yang, FENG Quan-xi, WEI Yan-ting, YUAN Sai-nan

(1 Big Data Analysis Technology and Algorithm Research Center, Guilin University of Technology, Guilin 541006, China; 2 College of Science, Guilin University of Technology, Guilin 541006, China)

Abstract: In order to solve constrained optimization problems, a novel differential evolution algorithm is proposed. Firstly, the algorithm uses two mutation strategies to balance local search ability and global search ability. Secondly, the new population is selected based on the feasibility-based rule, and poor individuals in the new population are replaced by the substitution strategy. Then, a mutation mechanism is used to mutate the worst individual to maintain the diversity of the population. Finally, the adaptive parameter control mechanism is introduced to enhance the robustness and adaptability of the algorithm. In order to verify the effectiveness of this algorithm, 10 benchmark constraint optimization problems and 10 engineering constraint optimization problems are tested. Experimental results show that the algorithm has high precision, fast convergence speed, and strong robustness.

Key words: differential evolution algorithm; constrained optimization; parameter self-adaptive; feasibility rule

1 引言

约束优化问题 (Constrained Optimization Problems, COPs) 是数学应用和工程实践中的常见问题. 令变量 $X = (x_1, x_2, \dots, x_D) \in S$, S 为可行域, D 是向

量 X 的维数. 约束优化问题的数学模型如下:

$$\begin{aligned} & \min f(X) \\ & \text{s. t. } g_i(X) \leq 0, i = 1, 2, \dots, m; \\ & h_j(X) = 0, j = m + 1, m + 2, \dots, n; \\ & L_d \leq x_d \leq U_d, d = 1, 2, \dots, D. \end{aligned} \tag{1}$$

收稿日期: 2018-11-20; 修回日期: 2018-12-07

基金项 目: 国家自然科学基金 (61763008; 11661030; 11401357; 11861027); 广西区自然科学基金项目 (2018GXNSFAA138095); 防城港市科学技术攻关项目 (防财教[2014]42 号)

式中,以 $[L_d, U_d]$ 的边界构成的 D 维实数空间 Ω ,称为决策空间,且 $S \subseteq \Omega$. $f(X)$ 为目标函数; $g_i(X)$, $i = 1, 2, \dots, m$ 为不等式约束函数; $h_j(X)$, $j = m+1, m+2, \dots, n$ 为等式约束函数.

为了平衡约束优化问题的可行解与不可行解, Wang Yong 等^[1-2]提出了适应性平衡模型(ATM, AATM)的约束处理机制. 2010年,为了克服单一约束处理方法的不足, Mallipeddi 和 Suganthan 提出了一种集成约束处理技术(ECHT)^[3]来求解 CEC2006^[4]的约束优化问题集. 2016年, Wang Yong 提出了一种结合了目标函数信息的可行规则约束进化优化方法(FROFI)^[5].

差分进化(Differential Evolution, DE)算法,由 Storn 和 Price^[6]在 1996 年首先提出. 2006 年, Brest J 提出了一种自适应参数控制的差分进化(jDE)^[7]算法. 2017 年, Mohamed^[8]提出了一种鲁棒性较好的三角形变异差分进化(NDE)算法. 2018 年, 赵齐辉等人^[9]提出了基于差分进化的蜻蜓算法, 同一年 Wang Bing-chuan 等人^[10]提出了一种能处理约束优化问题的复合差分进化算法 C²oDE(CCoDE).

约束差分进化算法改进需处理好全局与局部搜索的平衡. 由于一般差分进化算法的变异策略注重全局搜索而忽略了局部搜索, 所以本文引入可以加强局部搜索的三角变异算子, 并采用参数自适应控制方法, 提出一种新的自适应约束差分进化算法(记为 SaCDE), 算法设计了替换策略加快算法的收敛速度, 使用多样性保持机制增加种群多样性.

2 差分进化算法的描述

差分进化算法是一种启发式随机搜索算法, 一般的差分进化算法主要分 6 步, 依次是种群初始化、变异、交叉、评价、选择和终止条件判断. 算法结构简单, 容易编程实现. 且该算法的求解精度较高, 具有较强的全局搜索能力, 被广泛用于实际优化问题求解. 令差分进化算法的种群的规模为 NP, 最常用的种群初始化方法是均匀随机初始化. 差分进化算法中, 令 U 表示变异向量, 常用的变异策略如下:

$$\begin{aligned} & \text{DE}\backslash\text{rand}\backslash 1: \\ & U = X_{r_1} + F * (X_{r_2} - X_{r_3}) \end{aligned} \quad (2)$$

$$\begin{aligned} & \text{DE}\backslash\text{rand}\backslash 2: \\ & U = X_{r_1} + F * (X_{r_2} - X_{r_3}) + F * (X_{r_4} - X_{r_5}) \end{aligned} \quad (3)$$

$$\begin{aligned} & \text{DE}\backslash\text{best}\backslash 1: \\ & U = X_{\text{best}} + F * (X_{r_1} - X_{r_2}) \end{aligned} \quad (4)$$

$$\text{DE}\backslash\text{current to best}\backslash 1:$$

$$\begin{aligned} U &= X_{\text{current}} + F * (X_{\text{best}} - X_{\text{current}}) + \\ & F * (X_{r_1} - X_{r_2}) \end{aligned} \quad (5)$$

式中, F 为缩放因子, F 取值通常在 $[0, 1]$ 之间; r_1, r_2, r_3, r_4, r_5 , current 和 best 互不相等. 变异结束后, 对违反决策变量边界约束的个体进行边界处理, 最常用边界处理方法是该个体违反边界的维度重新赋值. 差分进化算法的常用交叉算子是二项交叉算子. 差分进化算法的选择机制是从父代个体与子代个体两两比较中选最好的个体.

3 自适应约束差分进化算法

本文提出的自适应约束差分进化算法主要从变异策略和加速收敛上进行改进, 同时引入种群多样性保持和参数自适应控制加强算法性能. 首先, SaCDE 算法使用混合变异策略, 利用 DE\rand\1 进行全局搜索, 利用三角形变异策略加强局部搜索. 算法执行过程中, 每一代的每一个个体的变异策略都从以上两种变异策略中以相同概率随机选出, DE\rand\1 变异策略的缩放系数 F 采用参数自适应进行控制. 其次, SaCDE 算法使用二项式交叉方案, 交叉概率 Cr 同样使用参数自适应机制进行控制. 然后, 算法使用可行规则方式选择较好个体, 得到一个新种群. 接着, SaCDE 算法采用替换策略替换掉较差个体, 以加快算法收敛. 最后算法引入多样性保持机制, 对最差个体的某些维度进行重新初始化变异, 以保持种群多样性.

3.1 变异策略

经典的 DE\rand\1 变异策略具有非常好的全局搜索能力, 但是局部搜索不足, 并且进化后期种群多样性会变差. 因此, 本文将一种可以加强局部搜索的三角形变异算子^[8](记为 DE\rand\triangular)与 DE\rand\1 变异算子结合进行混合变异.

第一个变异策略 DE\rand\1, 它的缩放因子 F 由参数自适应控制^[2]方式得到.

第二个变异策略 DE\rand\triangular 如下:

从父代种群 Pop 中随机选择 3 个个体, 将这些个体按照目标函数值和约束违反度由好到差排序, 依次为 $P_{\text{best}}, P_{\text{better}}, P_{\text{worst}}$. 令 $r_1 = 1, r_2 = \text{rand}(0.75, r_1), r_3 = \text{rand}(0.5, r_2), \text{rand}(c_1, c_2)$ 为 $[c_1, c_2]$ 间的均匀随机抽样函数. 令 $s = r_1 + r_2 + r_3$, 计算权重 $w_1 = r_1/s, w_2 = r_2/s, w_3 = r_3/s$. 按照权重 w_1, w_2, w_3 求出 3 个向量的加权平均, 即计算基础向量 X_{base} :

$$X_{\text{base}} = \omega_1 * P_{\text{best}} + \omega_2 * P_{\text{better}} + \omega_3 * P_{\text{worst}}$$

(6)

从而得到三角变异策略 DE\rand\triangular:

$$U = X_{\text{base}} + F_1 * (P_{\text{best}} - P_{\text{better}}) + F_2 * (P_{\text{best}} - P_{\text{worst}}) + F_3 * (P_{\text{better}} - P_{\text{worst}})$$

(7)

式中,变异系数 F_1 、 F_2 和 F_3 为 $[0, 1]$ 之间的均匀随机数. 对于每一代每一个个体,皆从以上两种变异机制中按相同概率随机选择一种变异策略进行变异. 三角形变异策略可以加强算法的局部搜索能力,DE\rand\triangular 与 DE\rand\l 变异策略的结合,使算法在全局搜索与局部搜索上达到了较好的平衡.

3.2 替换机制

使用 Deb 可行性规则选出新种群 pop 后,将试验种群中没有成功进化的个体记录到一个临时种群 TempPop 中,记 TempPop 种群大小为 N_{TP} ,从 $[1, N_{\text{TP}}]$ 中产生一个随机正整数 m . 令 TempPop 中最差个体为 tp_{worst} , pop 中比 tp_{worst} 差的个体的数为 n_{worst} ,若 $m \leq n_{\text{worst}}$,则使用 TempPop 中较好的 m 个个体替换 pop 中较差的 m 个个体,否则替换所有 n_{worst} 个个体. 将此替换机制记为算法 1,其伪代码见图 1. 采用试验个体替换机制可以快速淘汰劣解,保留搜索到的较好个体,并使算法快速收敛.

```
% Algorithm 1
N_TP = length(TempPop); % 临时种群规模
rn=randi(N_TP,1); % 产生随机正整数
count=1;
while ~isempty(TempPop) && count<=rn % 循环条件
    [tp_best,Idx_best]=FindBest(TempPop); % 求最好个体
    [p_worst,Idx_worst]=FindWorst(pop); % 求最差个体
    if BetterThan(tp_best,p_worst) % 若 tp_best 比 p_worst 好
        pop(Idx_worst)=TempPop(Idx_best); % 执行替换
        TempPop(Idx_best)=[];
        count=count+1; % count 累加
    else
        break; % 跳出循环
    end
end
```

图 1 试验个体替换机制的伪代码

3.3 种群多样性保持

为了保持种群多样性,本文选取最差个体进行变异. 若最差个体是不可行解,就对其部分维度实施均匀初始化变异,而对于可行解情形,则以 0.5 的概率对其部分维度进行均匀初始化变异. 变异的维度均为随机选择的 R 个维度,其中 R 是 $[1, D]$ 上的随机正整数.

3.4 参数自适应控制

本文使用自适应参数控制方法^[2],第 $t+1$ 代种群的第 k 个个体的缩放系数 $F_{t+1,k}$ 和交叉概率 $Cr_{t+1,k}$ 的计算公式如下:

$$F_{t+1,k} = \begin{cases} 0.1 + \text{rand}_1 * 0.9, & \text{if } \text{rand}_2 < 0.1 \\ F_{t,k}, & \text{otherwise} \end{cases}$$
$$Cr_{t+1,k} = \begin{cases} \text{rand}_3, & \text{if } \text{rand}_4 < 0.1 \\ Cr_{t,k}, & \text{otherwise} \end{cases}$$

(8)

式中,rand 为 $[0, 1]$ 间的随机数; F 的初始值为 0.5; Cr 的初始值为 0.9. 采用参数自适应控制机制能使算法较好地求解不同测试问题.

3.5 算法流程

本文提出的 SaCDE 算法的流程如图 2.

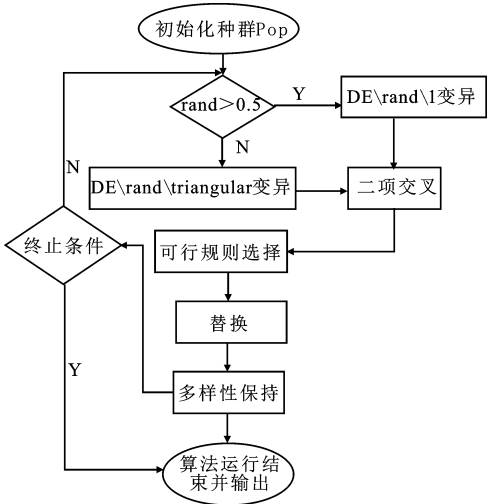


图 2 SaCDE 算法流程图

4 算法测试与分析

为了验证本文提出算法的有效性,将 SaCDE 与其它算法在两类约束优化问题上进行对比测试,选择 10 个标准约束优化问题^[4]和 10 个工程约束优化问题^[8]作为测试问题. 本文 SaCDE 算法的种群大小设置为 100. 对每一个问题,算法独立运行 30 次. 用 30 次运行结果的最好值 (MinBest)、平均值 (Min-Mean) 和标准差 (Std) 作为评价指标. 再根据这些数值结果的指标进行统计分析,本文用到 Mann-Whitney 秩和检验、Friedman 检验和 Wilcoxon 符号秩检验三种统计分析方法.

4.1 标准约束优化问题数值仿真与结果分析

算法对 10 个标准测试问题的最大评价次数 MaxFEs 设为 500 000 次^[10],对比算法为 ATM^[1]、CCoDE^[10]、ECHTEP^[3]、ES 和 FROFI^[5]. 其中 ES

算法的种群规模设置为 100, ATM、CCoDE、ECHTEP 和 FROFI 算法的种群规模等参数设置与原论文相同. 运行算法源代码测试 SaCDE 与 5 个对比算法,得到的数值结果如表 1,其中,NaN 表示没有找到相应最优值. 为了比较同一个问题下不同算法的差异性,根据 Mann-Whitney 秩和检验,表 1 中数值结果字体倾斜并加粗的表示本文提出的 SaCDE 算法的求解结果,正常字体显示的表示对比算法比 SaCDE 求解的数值结果差,带加粗字体标记的表示 SaCDE 的数值结果与其他算法结果相似,字体加粗并且单元格带灰色底纹标记的表示该算法的数值结果比 SaCDE 算法的好. 此外, Mann-Whitney 秩和检验的显著性水平设为 0. 05,比较结果如表 2,其中‘+’表示 SaCDE 的数值结果胜出的问题个数(比对比算法求解结果好),‘≈’表示 SaCDE 与对比算法结果相近(求解结果相似)的问题个数,‘■’表示 SaCDE 差(求解结果比对比算法差)的问题个数. 图 3 和图 4 分别是部分标准测试问题的收敛图与箱形图, $f(x^*)$ 表示当前已知最优值.

从表 1 可知,除 ES 算法外, SaCDE、ATM、

CCoDE、ECHTEP 和 FROFI 算法都能够 100%求得所有问题的近似最优解. SaCDE 求解 g01、g04、g06、g08、g09、g11、g12 的数值结果均达到或优于算法 ATM、CCoDE、ECHTEP、FROFI、ES 的数值结果. SaCDE 求解 g02 的数值结果比 ES 的数值结果好,和 ATM 求解结果相近,比 CCoDE、ECHTREP 和 FROFI 的数值结果稍差. 对于 g07 和 g10,算法的数值结果比 ATM、CCoDE、ECHTREP 和 ES 的数值结果好,稍差于 FROFI 的数值结果.

由表 2 Mann-Whitney 秩和检验结果比较可以知道, SaCDE 算法与 ATM、CCoDE、ECHTEP、ES 和 FROFI 算法相比,求解的数值结果胜出的问题个数分别为 6、7、7、10 和 1,数值结果相近的问题个数分别为 4、2、2、0 和 6,数值结果较差的个数分别为 0、1、1、0 和 3.

为了进一步比较所有算法在不同测试问题上的差异性,对平均值 (MinMean) 和最好值 (MinBest) 进行 Friedman 检验. 检验算法求解结果的显著性水平设置为 0. 05,检验结果见表 2.

表 1 标准约束优化问题的实验结果

算法	统计	g01	g02	g04	g06	g07
SaCDE	MinBest	−15	−0. 803 62	−30 665. 539	−6 961. 8139	24. 306 21
	MinMean	−15	−0. 788 54	−30 665. 539	−6 961. 813 9	24. 306 21
	Std	0	0. 011 488	1. 0914E-11	1. 819E-12	4. 64E-08
ATM	MinBest	−15	−0. 803 55	−30 665. 539	−6 961. 813 9	24. 306 65
	MinMean	−15	−0. 792 01	−30 665. 539	−6 961. 813 9	24. 314 81
	Std	0	0. 012 69	1. 029E-11	1. 819E-12	0. 011 764
CCoDE	MinBest	−14. 999 4	−0. 803 62	−30 665. 406	−6 961. 758 7	24. 309 87
	MinMean	−14. 998 6	−0. 802 77	−30 665. 192	−6 961. 445 5	24. 313 66
	Std	0. 000 893	0. 002 643	0. 109 002 04	0. 145 614 33	0. 001 901
ECHTEP	MinBest	−15	−0. 803 61	−30 665. 539	−6 961. 813 9	24. 306 21
	MinMean	−15	−0. 799 44	−30 665. 539	−6 961. 8139	24. 306 36
	Std	4. 52E−07	0. 005 294	1. 0095E-11	1. 819E-12	0. 000 21
ES	MinBest	−12. 246 6	−0. 615 54	−30 611. 186	−6 900. 486 6	52. 860 63
	MinMean	NaN	−0. 500 25	−3 0491. 395	− 5982. 213 2	290. 172 8
	Std	NAN	0. 045 109	112. 553 439	820. 918 716	487. 772 6
FROFI	MinBest	−15	−0. 803 62	−30 665. 539	−6 961. 813 9	24. 30621
	MinMean	−15	−0. 803 62	−30 665. 539	−6 961. 813 9	24. 306 21
	Std	0	1. 72E-07	1. 0914E-11	1. 819E-12	7. 31E-15

表 1(续) 标准约束优化问题的实验结果

算法	统计	g08	g09	g10	g11	g12
SaCDE	MinBest	−0.095 83	680.630 1	7 049.248	0.749 9	−1
	MinMean	−0.095 83	680.630 1	7 049.248	0.749 9	−1
	Std	2.83E-17	4.31E-13	6.43E-06	1.11E-16	0
ATM	MinBest	−0.095 83	680.630 2	7 053.902	0.749 949	−1
	MinMean	−0.095 83	680.641 5	7 236.579	0.750 25	−0.999 99
	Std	2.6E-17	0.014 003	105.099 3	0.000 23	7.42E-05
CCoDE	MinBest	−0.095 83	680.630 8	7 066.3	0.749 922	−1
	MinMean	−0.095 83	680.631 2	7 083.872	0.949 851	−1
	Std	2.74E-17	0.000 3	10.807 08	0.099 953	0
ECHTEP	MinBest	−0.095 83	680.630 1	7 049.248	0.749 9	−1
	MinMean	−0.095 83	680.630 1	7 049.249	0.749 901	−1
	Std	2.1E-17	1.57E-05	0.000 928	3.42E-06	0
ES	MinBest	−0.095 83	685.925 9	7 576.307	0.749 926	−1
	MinMean	−0.095 7	735.417 3	NaN	0.7896 29	−0.999 45
	Std	0.000 376	82.141 89	NaN	0.066 142	0.001 979
FROFI	MinBest	−0.095 83	680.630 1	7 049.248	0.749 9	−1
	MinMean	−0.095 83	680.630 1	7 049.248	0.749 9	−1
	Std	2.78E-17	4.61E-13	4.15E-12	1.11E-16	0

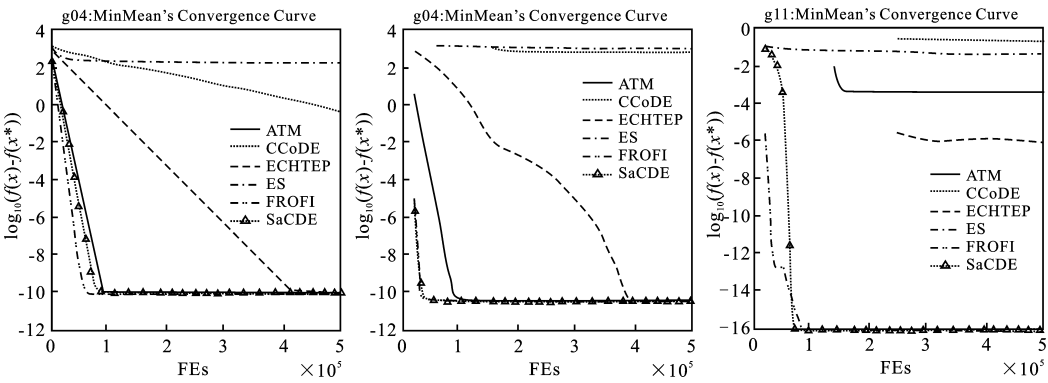


图 3 标准约束优化问题的半对数收敛图

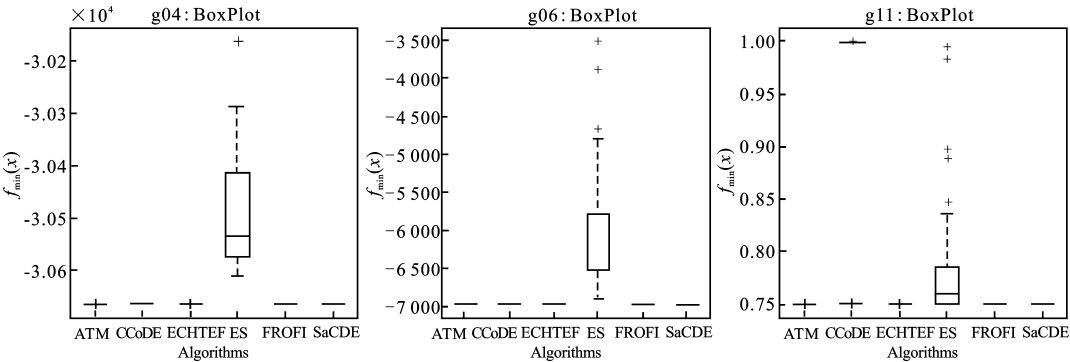


图 4 标准约束优化问题的箱形图

由表 2 Friedman 检验可知,对平均值来说,FROFI 的秩最小,SaCDE 算法的秩为 2.45,排列第二;而对最好值来说,SaCDE 的秩最小,为 1.95,FROFI 的秩排列第二;在这部分比较中,SaCDE 算法与 FROFI 算法表现持平,均优于其余 4 个算法.

表 2 标准约束优化问题的统计检验结果比较

Statistic Test	Symbol	SaCDE vs	ATM	CCoDE	ECHTEP	ES	FROFI
Mann-Whitney's Test	Win	+	6	7	7	10	1
	Tied	≈	4	2	2	0	6
	Lose	■	0	1	1	0	3
Friedman's Test	MinMean's Rank	2.45	3.80	4.05	2.95	5.90	1.85
	MinBest's Rank	1.95	3.65	4.25	3.05	5.90	2.20
Wilcoxon's Test	Signed Rank	R ⁻	3	4	6	0	6
		R ⁺	52	51	49	55	49

为了更进一步比较两个算法在不同问题上的差异性. 选择包含信息量较多的平均值 (MinMean) 作为样本, 利用 Wilcoxon 符号秩检验, 将 SaCDE 算法分别与其它对比算法进行配对检验. 检验算法的显著性水平设置为 0.05, 符号秩检验结果如表 2.

根据表 2 Wilcoxon 符号秩检验结果, 比较不同算法的符号秩和可以知道, 在 SaCDE 与 5 个算法的对比中, 符号秩 R⁻ 均小于 R⁺, 这表明 SaCDE 算法优于其它 5 个对比算法.

综合以上分析结果可以知道, 在 10 个标准约束优化问题上, SaCDE 算法和 FROFI 算法的性能相近, 并且这两个算法都优于 ATM、CCoDE、ECHTEP 及 ES 算法.

4.2 工程约束优化问题数值仿真与结果分析

为了进一步验证 SaCDE 算法的有效性, 本文将

SaCDE 算法与 AATM^[2]、ECHTDE^[3]、GA、NDE^[8]和 PSO 这 5 个算法在 10 个工程约束优化问题上进行对比测试. 其中 AATM、ECHTDE 的种群规模设置来自原论文, NDE、GA 和 PSO 算法的群体规模设置与 SaCDE 算法相同, 为 100. 由于这 10 个工程约束优化问题在过去的文献中缺乏统一测试, 不同算法对每个问题的最大评价次数设置不同, 为了便于比较, 本文将所有算法对工程优化问题求解的最大评价次数 MaxFEs 统一设置为 240 000 次, 其余参数与原论文相同. 运行算法源代码得到的测试结果如表 3, 表中数据标记与 4.1 节所述相同. 表 3 是工程约束优化设计问题的实验结果. 三种统计分析方法的检验结果比较见表 4.

表 3 工程约束优化问题的实验结果

算法	统计	拉压弹簧设计	焊接梁设计	压力容器设计	减速器设计	三杆桁架设计
SaCDE	MinBest	0.012 67	1.724 852	6 059.714 3	2 994.471 1	263.895 8
	MinMean	0.012 67	1.724 852	6 075.218 5	2 994.471 1	263.895 8
	Std	6.9E-18	1.11E-15	56.064 593	1.819E-12	0
AATM	MinBest	0.012 67	1.724 852	6 059.714 3	2 994.471 1	263.895 8
	MinMean	0.012 68	1.724 852	6 069.985	2 994.471 1	263.896 0
	Std	3.9E-05	4.24E-15	14.524 853	1.819E-12	0.000 249
ECHTDE	MinBest	0.012 67	1.724 852	6 059.714 3	2 994.471 1	263.895 8
	MinMean	0.012 67	1.724 852	6 059.714 3	2 994.471 1	263.895 8
	Std	7.2E-18	1.11E-15	9.095E-13	1.819E-12	0
GA	MinBest	0.0127 5	1.787 506	6 478.069 2	3 012.556	263.896 1
	MinMean	0.013 28	1.864 016	7 716.844 2	3 034.439 4	263.946 3
	Std	0.000 34	0.039 999	576.317 69	7.155 5098	0.085 644
NDE	MinBest	0.012 67	1.724 852	6 059.714 3	2 994.471 1	263.895 8
	MinMean	0.012 67	1.7248 52	6 061.768 5	2 994.471 1	263.895 8
	Std	1.02E-14	1.11E-15	7.685 829 9	1.819E-12	0
PSO	MinBest	0.012 94	1.761 726	6 067.955 9	3 020.292 5	263.895 8
	MinMean	0.013 76	1.846 294	6 243.188 3	3 047.192 1	263.895 8
	Std	0.000 5	0.047 335	169.800 55	15.116 076	0

表 3(续) 工程约束优化问题的实验结果

算法	统计	推力轴承设计	锥形轮设计	滚动轴承设计	蝶形弹簧设计	齿轮系设计
SaCDE	MinBest	1 625. 442 8	14. 672 53	−81 845. 256 9	1. 979 675	2. 7E-12
	MinMean	1 625. 442 8	14. 672 53	−72 643. 544	1. 983 829	1. 02E-11
	Std	4. 547E-13	1. 28E-14	16 679. 521 1	0. 022 37	9. 82E-12
AATM	MinBest	1 770. 783 0	14. 672 53	−81 845. 256 9	1. 979 969	2. 7E-12
	MinMean	2 132. 469 6	14. 788 24	−77 860. 419 65	1. 992 722	5. 68E-11
	Std	299. 862 23	0. 126 196	4 655. 421 877	0. 014 306	1. 94E-10
ECHTDE	MinBest	1 625. 442 8	14. 672 53	−81 845. 256 9	1. 979 675	2. 7E-12
	MinMean	1 625. 442 8	14. 672 53	−81 845. 256 76	1. 979 675	4. 06E-12
	Std	4. 547E-13	9. 43E-14	0. 000 426 034	1. 33E-15	5. 08E-12
GA	MinBest	2 665. 693 6	NaN	NaN	2. 200 735	2. 7E-12
	MinMean	3 380. 786 1	NaN	NaN	2. 349 212	2. 7E-12
	Std	450. 649 81	NaN	NaN	0. 092 108	1. 21E-27
NDE	MinBest	1 625. 442 8	14. 672 53	−81 845. 256 9	1. 979 675	2. 7E-12
	MinMean	1 625. 442 8	14. 672 53	−81 845. 256 9	2. 008 753	1. 76E-11
	Std	4. 547E-13	1. 58E-14	0	0. 052 709	5. 44E-11
PSO	MinBest	1 625. 452 7	14. 932 13	NaN	2. 184 691	2. 7E-12
	MinMean	1 712. 034 7	NaN	NaN	2. 564 383	1. 51E-08
	Std	100. 653 66	NaN	NaN	0. 176 774	7. 94E-08

表 4 工程约束优化问题的统计检验结果比较

Statistic Test	Symbol	SaCDE vs	ATM	CCoDE	ECHTEP	ES	FROFI
Mann-Whitney’s Test	Win	+	7	1	9	2	9
	Tied	≈	3	7	0	7	1
	Lose	—	0	2	1	1	0
Friedman’s Test	MinMean’s Rank	2. 45	3. 85	1. 95	5. 10	2. 50	5. 15
	MinBest’s Rank	2. 40	3. 55	2. 55	5. 50	2. 25	4. 75
Wilcoxon’s Test	Signed Rank	R [−]	16	14	1	9	0
		R ⁺	39	41	54	46	55

由表 3 可以知道, SaCDE 算法对所有工程问题都能 100%求得可行解. SaCDE 求解拉压弹簧设计、焊接梁设计、减速器设计、三杆桁架设计、推力轴承设计、锥形轮设计、蝶形弹簧设计的数值结果均达到或优于算法 AATM、ECHTDE、GA、NDE 和 PSO 的求解结果. SaCDE 求解压力容器设计的数值的结果比 GA 和 PSO 的求解结果好, 与 AATM 和 NDE 的数值结果相近, 比 ECHTDE 的数值结果差. 求解滚动轴承设计的结果比 AATM、GA 和 PSO 的数值结果好, 与 ECHTDE 的数值结果相近, 差于 NDE 的求解结果. 求解齿轮系设计的结果比 AATM 和 PSO 的数值结果好, 与 NDE 求解结果相近, 比 ECHTDE 和 GA 的数值结果稍差.

由 Mann-Whitney 秩和检验可以知道, SaCDE

算法与 AATM、ECHTDE、GA、NDE 和 PSO 算法相比, 数值结果胜出的个数分别为 7、1、9、2、9, 结果相近的问题个数分别为 3、7、0、7、1, 结果较差的问题个数为分别为 0、2、1、1、0.

由 Friedman 检验可知, 在 MinMean 上, ECHTDE 的秩最小, SaCDE 排第二, 为 2. 45; 在 MinBest 上, NDE 的秩最小, SaCDE 排第二, 为 2. 40. 这部分比较中 SaCDE 稍差于 ECHTDE 和 NDE.

由 Wilcoxon 符号秩检验可知, 符号秩和 R[−] 均小于 R⁺, 在这部分比较中, SaCDE 的数值结果均好于对比算法, 性能表现最好.

综合以上分析结果可以知道, 在 10 个工程约束优化问题上, SaCDE 算法的求解效果与 ECHTDE

相近,且均优于 AATM、NDE、GA 和 PSO 这 4 个算法,算法的性能较好。

5 结束语

本文利用混合变异策略使算法在全局搜索和局部搜索上达到了较好的平衡。SaCDE 的试验个体替换策略克服了基于三角形变异的算法收敛速度慢的缺陷。此外 SaCDE 的参数自适应机制避免了对算法进行手动调参,这对求解复杂约束优化问题非常有效。本文在 SaCDE 中引入多样性保持机制,使种群在进化过程中保持良好多样性。根据仿真结果分析可以知道, SaCDE 算法在本文测试问题上比 ATM、CCoDE、ECHTEP、ES 算法及 AATM、GA、NDE、PSO 算法的求解效果好,与 FROFI 和 ECHTDE 的性能相近,显示出较高的求解精度、较快的收敛速度和较强的鲁棒性。本文的 SaCDE 算法在两类测试问题上总体表现不错,但对个别问题求解还存在稳定性不足的问题。所以接下来的工作需要继续提升算法鲁棒性,并使算法能求解更多的约束优化问题。

参考文献:

- [1] WANG Y, CAI Z, ZHOU Y, et al. An adaptive tradeoff model for constrained evolutionary optimization[J]. IEEE Transactions on Evolutionary Computation, 2008, 12(1): 80-92.
- [2] WANG Y, CAI Z, ZHOU Y. Accelerating adaptive trade - off model using shrinking space technique for constrained evolutionary optimization[J]. International Journal for Numerical Methods in Engineering, 2010, 77(11): 1501-1534.
- [3] MALLIPEDDI R, SUGANTHAN P N. Ensemble of constraint handling techniques[J]. IEEE Transactions on Evolutionary Computation, 2010, 14 (4): 561-579.
- [4] LIANG J J, RUNARSSON T P, MEZURA-MONTES E, et al. Problem definitions and evaluation cri-

teria for the CEC 2006 special session on constrained real-parameter optimization[J]. Journal of Applied Mechanics, 2006, 41(8): 8-31.

- [5] WANG Y, WANG B C, LI H X, et al. Incorporating objective function information into the feasibility rule for constrained evolutionary optimization[J]. IEEE Transactions on Cybernetics, 2016, 46 (12): 2938-2952.
- [6] STORN R, PRICE K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces[J]. Journal of Global Optimization, 1997, 11(4): 341-359.
- [7] BREST J, GREINER S, BOSKOVIC B, et al. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems [J]. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646-657.
- [8] MOHAMED A W. A novel differential evolution algorithm for solving constrained engineering optimization problems[J]. Journal of Intelligent Manufacturing, 2018, 29(3): 659-692.
- [9] 赵齐辉, 杜兆宏, 刘升, 等. 差分进化的蜻蜓算法[J]. 微电子学与计算机, 2018, 35(7): 101-105.
- [10] WANG B C, LI H X, LI J P, et al. Composite differential evolution for constrained evolutionary optimization[J]. IEEE Transactions on Systems Man & Cybernetics Systems, 2018, DOI: 10. 1109/TSMC. 2018. 2807785

作者简介:

栗朝阳 男, (1993-), 硕士研究生. 研究方向为计算智能、机器学习.

封全喜(通讯作者) 男, (1980-), 博士, 副教授. 计算智能、机器学习. E-mail: fqx9904@163.com.

韦彦婷 女, (1995-), 硕士研究生. 研究方向为计算智能、机器学习.

苑赛楠 女, (1995-), 硕士研究生. 研究方向为计算智能、机器学习.