文章编号: 1001-0920 (2016) 05-0790-07 DOI: 10.13195/j.kzyjc.2015.0487

基于一次指数平滑法的自适应差分进化算法

赵志伟[1,2], 杨景明[1], 呼子宇[1], 车海军[1]

(1. 燕山大学 a. 电气工程学院，b. 国家冷轧板带装备及工艺工程技术研究中心, 河北 秦皇岛 066004；2. 唐山学院 计算机科学与技术系，河北 唐山 063000)

摘 要: 提出一个策略和控制参数自适应的差分进化 (ESADE) 算法. ESADE 算法将指数平滑法和轮盘赌选择法结合到一起, 根据先前成功的经验在策略候选池中为每个个体自适应地选择变异策略来匹配进化的不同阶段. 在进化过程中, ESADE 算法使用柯西分布和正态分布为控制参数产生适当的值, 并使用指数平滑法进行自适应. 大量的仿真实验结果表明, ESADE 算法要优于其他差分进化算法.

关键词: 差分进化；指数平滑法；自适应；柯西分布；正态分布

中图分类号: TP18 **文献标志码:** A

Self-adaptive differential evolution algorithm based on exponential smoothing

ZHAO Zhi-wei[1,2], *YANG Jing-ming*[1], *HU Zi-yu*[1], *CHE Hai-jun*[1]

(1a. School of Electrical Engineering，1b. National Engineering Research Center for Equipment and Technology of Cold Strip Rolling，Yanshan University，Qinhuangdao 066004，China；2. Department of Computer Science and Technology，Tangshan University，Tangshan 063000，China．Correspondent：ZHAO Zhi-wei，E-mail：wzzwzz@sina.com)

Abstract: This paper presents a differential evolution algorithm based on exponential smoothing (ESADE), with self-adaptive strategy and control parameters, which employs single exponential smoothing and roulette wheel selection, and adaptively selects mutation strategy for each individual from the strategy candidate pool to match different stages of the evolution according to their previous successful experience. Cauchy distribution and normal distribution are used to generate appropriate values for control parameters during the evolutionary process, and single exponential smoothing method is used to realize self-adaption. A large amount of simulation experiments are made, and experimental results show that the ESADE is better than other DE algorithms.

Keywords: differential evolution；exponential smoothing；self-adaptation；Cauchy distribution；normal distribution

0 引 言

差分进化算法 (DE) 是由 Storn 等[1]开发的一种基于种群的随机搜索技术. 目前, 差分进化算法被应用到了很多领域, 比如机械工程、通信、模式识别、信号处理等[2-5].

标准差分进化算法需要预先设定好试验向量产生策略和控制参数值, 在进化过程中不能根据优化问题和进化阶段的不同适当调整变异策略和控制参数值. 因此, 标准差分进化算法在处理多峰和高维问题时很容易陷入局部最优, 出现早熟的现象.

为了改善标准差分进化算法的性能, 很多学者提出了改进方案. Brest 等[6]提出了 SADE 算法, 该算法根据特定的概率为下一代个体产生控制参数值或沿用上一代的控制参数值, 随着不断地进化而实现控制参数的自适应. Zou 等[7]提出了 MDE 算法, 该算法使用高斯分布和均匀分布产生随机数作为控制参数值, 并根据进化代数选择变异策略. Pan 等[8]提出了 SspDE 算法, 该算法将变异策略和控制参数值分配给每个个体, 经过一定代数后, 该算法根据一个较大的概率将产生更好个体的变异策略和控制参数值重新

收稿日期: 2015-04-19；**修回日期:** 2015-08-28.

基金项目: 国家自然科学基金钢铁联合基金项目(U1260203)；河北省高等学校创新团队领军人才培育计划项目(LJRC013)；国家冷轧板带及装备工程研究中心开放课题项目(2012005).

作者简介: 赵志伟(1980−), 男, 副教授, 博士生, 从事进化算法和轧制过程智能优化的研究；杨景明(1957−), 男, 教授, 博士生导师, 从事冶金机械综合自动化及人工智能等研究.

分配给每个个体, 或者随机分配变异策略和控制参数值. Mallipeddi 等[9]提出了 ESMDE 算法, 该算法根据当前种群构建了一个替代模型, 在进化的不同阶段产生更有竞争力的子代, 替代模型也会随着进化过程不断更新. Fan 等[10]提出了 DMPSADE 算法, 该算法为决策变量的每一维生成一个比例因子, 并为每个个体选择一个变异策略和生成一个交叉率.

上述研究成果主要集中在对控制参数和变异策略的研究上, 但在产生控制参数值和选择变异策略时过多地依赖随机生成, 很少考虑进化过程的历史信息, 故本文提出一种新型策略和控制参数自适应的差分进化 (ESADE) 算法. ESADE 算法将一次指数平滑法 (ES) 和轮盘赌选择法结合到一起, 根据历史经验在策略候选池中自适应地选择变异策略, 并使用带有自适应参数的均匀分布和柯西分布为每个个体产生合理的控制参数值. 为了验证 ESADE 算法的性能, 本文采用 CEC2005 标准测试函数[11]进行仿真实验, 实验结果表明, 本文算法的性能明显优于其他对比算法.

1　基于ES的自适应差分进化算法

对于不同的优化问题甚至是同一问题的不同进化阶段, 最适合的变异策略和控制参数值都是不同的[6]. 因此, 本文在标准差分进化算法的基础上提出了一个基于一次指数平滑法 (ES) 的自适应差分进化 (ESADE) 算法.

1.1　变异策略自适应

DE 算法的变异策略有很多, 有些策略有较强的探索能力, 而有些策略有较强的开发能力. 常见的变异策略如下[12-14]:

$$V_{i,G} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}), \tag{1}$$

$$V_{i,G} = X_{\mathrm{best},G} + F \times (X_{r1,G} - X_{r2,G}), \tag{2}$$

$$V_{i,G} = X_{i,G} + F \times (X_{\mathrm{best},G} - X_{i,G}) + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G}), \tag{3}$$

$$V_{i,G} = X_{\mathrm{best},G} + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G}), \tag{4}$$

$$V_{i,G} = X_{r5,G} + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G}). \tag{5}$$

其中: F 为变异比例因子; $X_{\mathrm{best},G}$ 为第 G 代的最优解; $r1$、$r2$、$r3$、$r4$ 和 $r5$ 为来自 [1,NP] 中的 5 个互不相等的整数. 上述 5 个变异策略, 即式 (1) ~ (5) 通常被称为 DE/rand/1、DE/best/1、DE/rand-to-best/2、DE/best/2 和 DE/rand/2.

为了能够实现对于不同问题或者同一问题的不同进化阶段都能选择出最适合的变异策略, ESADE 算法构建了一个变异策略候选池, 候选池由上述 5 个变异策略构成, 并引入基于指数平滑法 (ES) 和轮盘赌选择法的自适应机制. 具体的过程如下.

1) 根据每个变异策略被选择的概率 $K_{j,G}$ (因为候选池中共有 5 个变异策略, 所以初值 $K_{j,0}$ 为 0.2), 采用轮盘赌选择法为种群中的每个目标向量 $X_{i,G}$ 从策略候选池中选择变异策略, 并计算每个变异策略被选择的次数 s_j ($j = 1, 2, \cdots, 5$).

2) 种群经过变异和交叉操作后, 计算每个变异策略产生更好个体的次数 w_j ($j = 1, 2, \cdots, 5$).

3) 根据下式计算每个变异策略产生更好个体的概率:

$$R_{j,G} = \frac{w_j}{s_j},\ j = 1, 2, \cdots, 5, \tag{6}$$

其中 G 为当前代数.

4) 使用一次指数平滑法按下式预测每个变异策略在下一代产生更好个体概率:

$$R_{j,G+1} = a \times R_{j,G} + (1 - a) \times R_{j,G-1},\quad j = 1, 2, \cdots, 5. \tag{7}$$

其中: G 为当前代数; a 为一个 (0,1) 间的正常数, 本文设置为 0.2.

5) 按下式更新下一代每个变异策略被选择的概率:

$$K_{j,G+1} = \frac{R_{j,G+1}}{\sum_{j=1}^{5} R_{j,G+1}},\ j = 1, 2, \cdots, 5, \tag{8}$$

其中 G 为当前代数.

6) s_j 和 w_j 置 0.

指数平滑法 (ES) 的引入实现了以进化过程的历史经验为依据, 预测每个变异策略在下一代进化中产生更好个体的概率 $R_{j,G+1}$. 从式 (7) 不难看出, 在经过 $1/a$ 代后, 历史经验对概率 $R_{j,G+1}$ 的影响程度将以指数形式减小到 $(1-a)^{1/a}$. 即正常数 a 越接近于 1, 历史经验对概率 $R_{j,G+1}$ 的影响程度下降越快; 正常数 a 越接近于 0, 历史经验对概率 $R_{j,G+1}$ 的影响程度下降越慢. 考虑到进化过程会有波动, 但长期趋势变化不会太大, 故本文正常数 a 取 0.2. 此外, 为了避免某个变异策略被选择的概率 $K_{j,G}$ 过大, 从而出现在下一代进化中过度选择该策略的情况, 本文引入了轮盘赌选择法来完成变异策略的选择, 这样既保证了选择概率大的策略被选择的可能性大, 又保证了策略选择具有一定的随机性.

1.2　控制参数自适应

ESADE 算法在控制参数自适应中引入了带有自适应参数的柯西分布随机数和正态分布随机数来产生控制参数值, 其中自适应参数根据产生更好个体的控制参数值使用指数平滑法 (ES) 预测下一代进化中

的自适应参数. 正态分布和柯西分布的引入保证了以较大的概率生成期望的控制参数值, 又不失随机性, 从而保证种群的多样性.

在变异和交叉操作中, 根据如下公式为每一个目标向量 $X_{i,G}$ 产生一个变异比例因子 $F_{i,G}$ 和一个交叉率 $\mathrm{CR}_{i,G}$:

$$F_{i,G} = N\mathrm{rand}(T_G, m), \tag{9}$$

$$\mathrm{CR}_{i,G} = C\mathrm{rand}(U_G, n). \tag{10}$$

其中: G 为当前代数; $N\mathrm{rand}()$ 为正态分布随机数; T_G 为自适应参数, 即正态分布的均值, 本文初始设置为 0.5; m 为正态分布的标准差, 本文设置为 0.1; $C\mathrm{rand}()$ 为柯西分布随机数; U_G 为自适应参数, 即柯西分布的位置参数, 本文初始设置为 0.5; n 为柯西分布的尺度参数, 本文设置为 0.1. 如果 $F_{i,G}$ 超出 (0,1), 则将其设置为 1; 如果 $\mathrm{CR}_{i,G}$ 超出 [0,1], 则将其设置为 0 或 1.

在每一代的进化中, 每一个目标向量 $X_{i,G}$ 都根据与其对应的变异因子 $F_{i,G}$ 和交叉概率 $\mathrm{CR}_{i,G}$ 产生一个对应的实验向量 $U_{i,G}$. 如果 $U_{i,G}$ 优于 $X_{i,G}$, 那么对应的 $F_{i,G}$ 和 $\mathrm{CR}_{i,G}$ 将被保存到集合 FB 和 CRB 中. 在这一代进化结束后, 变异因子 $F_{i,G}$ 和交叉概率 $\mathrm{CR}_{i,G}$ 的自适应参数 (即式 (9) 和 (10) 中的 T_G 和 U_G) 使用一次指数平滑法 (ES) 根据集合 FB 和 CRB 的信息进行更新, 以实现控制参数的自适应, 即自适应参数 T_G 和 U_G 按下式进行更新:

$$T_{j,G+1} = a \times T_{j,G} + (1-a) \times T_{j,G-1}, \tag{11}$$

$$U_{j,G+1} = a \times U_{j,G} + (1-a) \times U_{j,G-1}. \tag{12}$$

其中: G 为当前代数; a 为一个 (0,1) 间的正常数, 本文设置为 0.2. T_G 和 U_G 按下式计算:

$$T_{j,G} = \frac{\sum\limits_{F_{i,G} \in \mathrm{FB}} F_{i,G}}{M}, \tag{13}$$

$$U_{j,G} = \frac{\sum\limits_{\mathrm{CR}_{i,G} \in \mathrm{CRB}} \mathrm{CR}_{i,G}}{N}. \tag{14}$$

其中: M 为集合 FB 的元素个数, N 为集合 CRB 的元素个数. 更新结束后, 清空 FB 和 CRB.

1.3 ESADE 算法流程

ESADE 算法的主要流程描述如下.

Step 1: 设置种群大小 NP, 最大函数评价次数 MAX_FES, 概率值 $K_{j,0}$, 自适应参数 T_0 和 U_0, 正常数 a, 函数评价次数计数器 FES, 代计数器 G.

Step 2: 初始化种群 P_0, 并评估个体, 置 FES = NP; 创建计数器数组 s_j 和 w_j $(j = 1, 2, \cdots, 5)$.

Step 3: 置 $i = 1$.

Step 4: 根据 $K_{j,G}$ 采用轮盘赌选择法选择变异策略.

Step 5: 根据式 (9) 和 (10) 产生控制参数 $F_{i,G}$ 和 $\mathrm{CR}_{i,G}$.

Step 6: 根据 Step 5 中产生的 $F_{i,G}$、$\mathrm{CR}_{i,G}$ 和 Step 4 中选择的变异策略对目标向量 $X_{i,G}$ 进行变异和交叉操作.

Step 7: 如果 $f(U_{i,G}) \leqslant f(X_{i,G})$, 则 $X_{i,G+1} = U_{i,G}$, 并将 $F_{i,G}$、$\mathrm{CR}_{i,G}$ 加入对应集合 FB、CRB 中, $w_j = w_j + 1$; 否则 $X_{i,G+1} = X_G$.

Step 8: 令 $\mathrm{FES} = \mathrm{FES} + 1$, $i = i + 1$, 若 $i < \mathrm{NP}$, 则转到 Step 4.

Step 9: 计算每个变异策略被选择的次数 s_j, 并根据式 (6) ~ (8) 计算 $K_{j,G+1}$, 然后置 $s_j = 0$, $w_j = 0$; 根据式 (11) 和 (12) 计算 T_{G+1} 和 U_{G+1}, 并清空 FB 和 CRB.

Step 10: 令 $G = G + 1$. 若 $\mathrm{FES} \leqslant \mathrm{MAX\_FES}$, 则转至 Step 3.

Step 11: 输出最优解.

2 测试函数与参数设置

为了验证 ESADE 算法的性能, 选择 CEC2005 的前 12 个标准测试函数进行仿真实验. 其中: $F_1 \sim F_5$ 为单峰函数, $F_6 \sim F_{12}$ 为基本多峰函数. 这 12 个函数的性质覆盖了单峰、多峰、移位、可分、不可分、可扩展、不可扩展、旋转、带噪声等, 具体细节可参见文献 [11].

另外, 本文选择了标准 DE 算法及 3 个比较经典的自适应 DE 算法 (即 MDE、SADE 和 SspDE) 作为比较对象, 以深入研究 ESADE 算法的各项性能. 根据文献 [15-16] 的建议, 本文将标准 DE 算法的控制参数 F 和 CR 分别设置为 0.5 和 0.4. MDE、SADE 和 SspDE 算法的参数设置遵循原始文献. 对于 ESADE 算法, 本文将 a 设置为 0.2, m 和 n 均设置为 0.1. 所有算法的种群大小 NP 均为 100, 最大函数评价次数 MAX_FES 均为 100 000. 每个算法都要针对 30 维和 50 维标准测试函数独立运行 30 次, 采集各测试函数的均值和标准差, 并进行 Wilcoxon 符号秩检验, 用以对比不同算法在统计学意义上是否存在显著差异.

3 实验结果分析

3.1 ESADE 算法与对比算法的比较分析

表 1 和表 2 列出了 30 维和 50 维标准测试函数运行 30 次获得的目标函数值的均值和标准差. 在 30 维情况下, ESADE 获得了 9 个最好的均值和 6 个最好的标准差; 在 50 维情况下, ESADE 获得了 7 个最好的均值和 5 个最好的标准差. 这说明 ESADE 算法比其他算法更能逼近理论最优值, 并且有着很好的稳定性.

为了进一步分析实验结果, 本文使用 50 维标准

测试函数运行 30 次的结果绘制了盒图, 如图 1 所示.

从图 1 不难看出, ESADE 算法在大多数函数上的表现要优于其他算法, 并且很少产生异常点, 即便产生异常点, 也距离所有样本的均值较近. 此外, 从图中也能看出, ESADE 算法样本分布较其他算法集中, 再次说明 ESADE 算法的稳定性要高于其他算法.

表 1　30 维标准测试函数运行 30 次的均值和标准差

函数	ESADE		DE		MDE		SADE		SspDE	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
F_1	−4.50e+02	0.00e+00	−4.50e+02	6.78e-11	−4.50e+02	0.00e+00	−4.50e+02	1.00e-011	−4.50e+02	0.00e+00
F_2	−4.50e+02	0.00e+00	5.04e+03	1.10e+03	−4.50e+02	4.23e-002	4.72e+02	3.48e+02	−4.50e+02	1.46e-01
F_3	1.20e+05	7.89e+04	3.42e+07	6.54e+06	5.77e+05	2.44e+05	9.33e+06	5.12e+06	3.67e+05	2.12e+05
F_4	−4.42e+02	3.10e+01	1.20e+04	2.33e+03	−4.27e+02	3.44e+01	1.72e+03	8.21e+02	−3.31e+02	9.33e+01
F_5	1.01e+03	6.56e+02	3.08e+03	5.63e+02	1.30e+03	5.79e+02	1.04e+03	4.73e+02	1.98e+03	5.35e+02
F_6	3.90e+02	1.11e+00	4.16e+02	3.54e+01	4.04e+02	2.10e+01	4.11e+02	2.06e+01	4.44e+02	3.53e+01
F_7	−1.80e+02	1.00e-02	−1.79e+02	7.09e-02	−1.80e+02	1.14e-02	−1.80e+02	6.01e-02	−1.80e+02	1.73e-02
F_8	−1.20e+02	5.21e-02	−1.19e+02	6.01e-02	−1.19e+02	4.56e-02	−1.19e+0	5.22e-022	−1.19e+02	4.40e-02
F_9	−3.30e+02	2.12e-03	−2.28e+02	6.63e+00	−2.85e+02	1.10e+01	−2.76e+02	5.01e+00	−3.30e+02	1.68e-06
F_{10}	−2.52e+02	1.49e+01	−1.19e+02	1.05e+01	−2.79e+02	1.33e+01	−1.50e+02	9.50e+00	−2.91e+02	5.21e+00
F_{11}	1.24e+02	4.78e+00	1.23e+02	2.50e+00	1.26e+02	7.04e+00	1.27e+02	1.30e+00	1.17e+02	1.43e+00
F_{12}	3.11e+04	3.20e+04	4.89e+05	6.32e+04	5.69e+05	1.19e+05	2.49e+05	4.35e+04	6.89e+04	1.52e+04

表 2　50 维标准测试函数运行 30 次的均值和标准差

函数	ESADE		DE		MDE		SADE		SspDE	
	均值	标准差	均值	标准差	均值	标准差	均值	标准差	均值	标准差
F_1	−4.50e+02	0.00e+00	−4.49e+02	3.54e-04	−4.50e+02	1.73e-11	−4.50e+02	8.01e-06	−4.50e+02	1.55e-10
F_2	−4.49e+02	4.75e+00	3.49e+04	2.65e+03	−1.01e+02	2.35e+02	2.49e+04	5.80e+03	−9.26e+01	1.80e+02
F_3	1.14e+06	6.14e+05	2.39e+08	3.88e+07	2.59e+06	9.39e+05	6.49e+07	1.89e+07	2.89e+06	1.39e+06
F_4	4.19e+03	2.37e+03	6.80e+04	5.11e+03	4.79e+03	2.09e+03	3.79e+04	1.12e+04	8.78e+03	2.64e+03
F_5	6.19e+03	1.28e+03	1.09e+04	9.29e+02	5.17e+03	1.04e+03	6.00e+03	1.13e+03	5.80e+03	5.09e+02
F_6	4.03e+02	2.09e+01	1.28e+03	3.76e+02	4.49e+02	3.48e+01	4.74e+02	5.09e+01	7.69e+03	3.23e+02
F_7	−1.80e+02	1.15e-02	−1.78e+02	1.19e-01	−1.79e+02	2.48e-02	−1.79e+02	1.10e-01	−1.79e+02	1.49e-01
F_8	−1.19e+02	3.39e-02	−1.18e+02	3.60e-02	−1.18e+02	3.19e-02	−1.18e+02	2.32e-02	−1.18e-02	3.70e-02
F_9	−2.89e+02	6.19e+00	−7.78e+01	1.19e+01	−2.39e+02	1.89e+01	−1.69e+02	1.19e+01	−3.29e+02	1.15e+00
F_{10}	−1.50e+02	3.48e+01	9.69e+01	1.36e+01	−2.24e+02	8.23e+01	4.29e+01	9.49e+00	−1.89e+02	1.59e+01
F_{11}	1.60e+02	6.29e+00	1.60e+02	1.39e+00	1.59e+02	7.69e+00	1.59e+02	1.39e+00	1.45e+02	2.19e+00
F_{12}	7.09e+05	7.49e+04	2.69e+06	3.08e+05	2.88e+06	8.79e+05	1.53e+06	1.10e+05	4.47e+05	4.89e+04

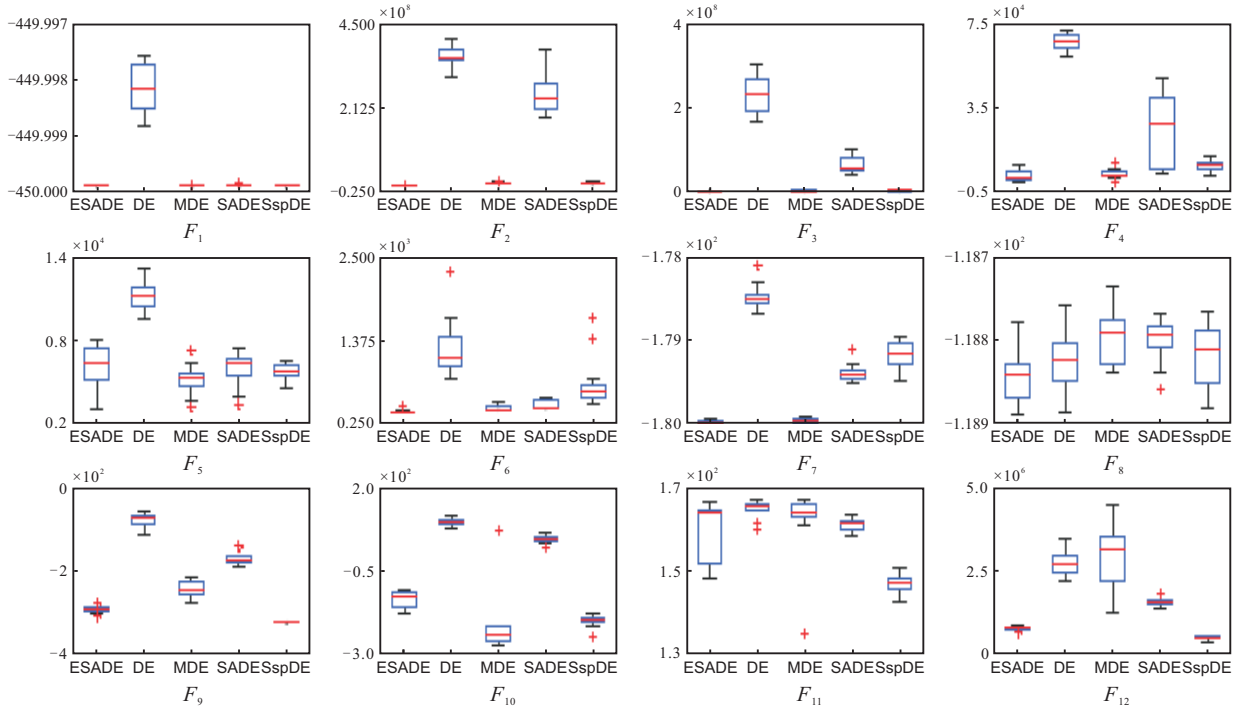

图 1　50 维标准测试函数运行 30 次的最优值盒图

表3和表4列出了每个算法和最好算法之间进行置信度为5%的Wilcoxon符号秩检验而得到的P-value. 本文中的最好算法被定义为获得最优均值的算法. 如果不只一个算法获得了最优均值, 那么标准差最小的那个就是最好算法.

表 3 30维标准测试函数运行30次的P-value

函数	ESADE	DE	MDE	SADE	SspDE
F_1	NA	1.697 8e-6	NA	6.470 5e-04	NA
F_2	NA	1.734 4e-6	1.734 4e-4	1.734 4e-4	1.734 4e-4
F_3	NA	1.734 4e-6	1.734 4e-4	1.734 4e-4	1.734 4e-4
F_4	NA	1.734 4e-6	1.500 1e-3	1.734 4e-4	1.734 4e-4
F_5	NA	1.734 4e-6	9.953 7e-2	8.210 1e-1	1.470 1e-2
F_6	NA	1.734 4e-6	1.734 4e-4	1.734 4e-4	1.734 4e-4
F_7	NA	1.734 4e-4	6.500 1e-1	4.100 1e-2	3.089 9e-2
F_8	NA	3.339 9e-1	3.950 1e-1	6.500 1e-1	6.910 1e-1
F_9	6.915 4e-2	1.734 4e-4	1.734 4e-4	1.734 4e-4	NA
F_{10}	1.734 4e-4	1.734 4e-4	1.730 2e-1	1.734 4e-4	NA
F_{11}	8.060 1e-4	1.734 4e-4	1.779 9e-3	1.734 4e-4	NA
F_{12}	NA	1.734 4e-4	1.734 4e-4	1.734 4e-4	1.060 2e-2

表 4 50维标准测试函数运行30次的P-value

函数	ESADE	DE	MDE	SADE	SspDE
F_1	NA	1.734 4e-6	6.340 6e-4	1.734 4e-6	1.734 4e-6
F_2	NA	1.734 4e-6	1.734 4e-6	1.734 4e-6	1.734 4e-6
F_3	NA	1.734 4e-6	1.210 2e-3	1.734 4e-6	1.210 2e-3
F_4	NA	1.734 4e-6	4.270 5e-1	1.734 4e-6	1.734 4e-6
F_5	6.920 1e-2	1.734 4e-6	NA	1.991 7e-2	6.908 7e-2
F_6	NA	1.734 4e-6	9.867 9e-4	9.867 9e-4	1.734 4e-6
F_7	NA	1.734 4e-6	3.562 1e-2	1.734 4e-6	1.734 4e-6
F_8	NA	1.397 6e-1	5.385 7e-3	2.610 1e-3	1.251 5e-1
F_9	1.734 4e-6	1.734 4e-6	1.734 4e-6	1.734 4e-6	NA
F_{10}	1.061 7e-2	1.734 4e-6	NA	8.048 9e-4	1.461 2e-2
F_{11}	1.734 4e-6	1.734 4e-6	8.049 9e-4	1.734 4e-6	NA
F_{12}	1.734 4e-6	1.734 4e-6	1.734 4e-6	1.734 4e-6	NA

在表3和表4中, 本文用NA表示最好算法. 如果P-value小于0.05, 则零假设被拒绝, 也就是说最好算法在统计学意义上要明显优于当前算法; 否则零假设被接受, 也就是说当前算法和最好算法在统计学意义上没有明显差别. 在表3中, ESADE算法是9个函数的最好算法, 并且有1个函数的P-value大于0.05. 也就是说, 在30维情况下, ESADE算法在12个函数中有9个函数获得了最好的结果, 且有1个函数的结果和最好算法的结果没有差别. 在表4中, ESADE算法是7个函数的最好算法, 并且有1个函数的P-value大于0.05. 同样, 在50维的情况下, ESADE算法在12个函数中有7个函数获得了最好的结果, 并且有1个函数的结果和最好算法的结果没有差别. 可见, 在统计学意义上, ESADE算法有着比其他算法更好的表现.

3.2 ESADE算法的收敛性分析

此外, 为了研究ESADE算法的收敛性能, 本文使用50维标准测试函数的实验结果绘制了如图2所示的收敛曲线. 为了便于分析, 图2采用了对数坐标系, 即纵轴为30次目标函数值的均值与理论最优值之差的对数值, 横轴为函数评价次数.

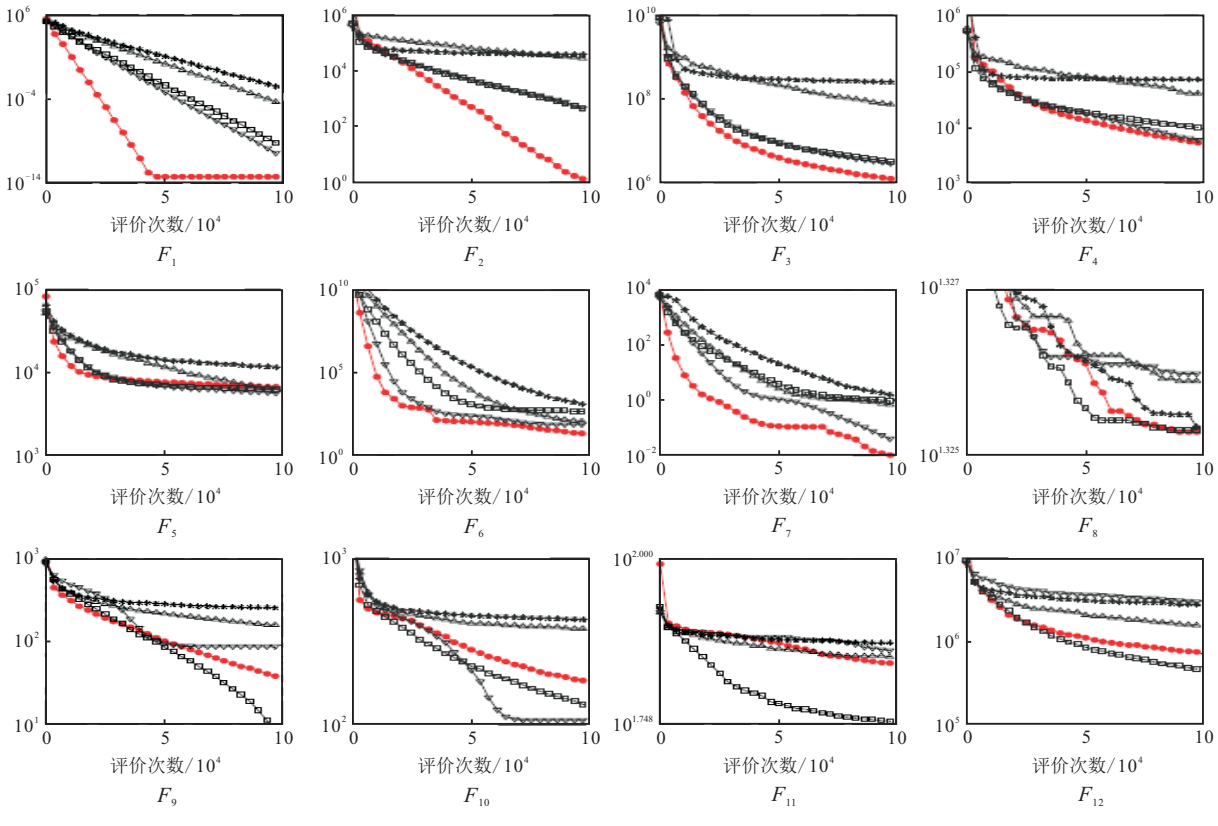

图 2 50维标准测试函数运行30次的收敛曲线

从图 2 可以看出, 在大部分函数上, ESADE 的算法的收敛速度要比其他算法快, 主要原因有 3 个: 1) 本算法能够自适应地选择最适合的变异策略来匹配当前寻优问题和进化阶段的特点, 从而加快进化速度; 2) ESADE 算法的策略候选池中包含了 DE/best/1、DE/rand-to-best/2 和 DE/best/2, 这几个策略对于单峰和多峰函数有着很好的收敛效果, 并且将最好解的信息整合了进来, 从而大幅度提高了 ESADE 算法的收敛速度; 3) ESADE 算法能够根据历史经验产生当前进化中最适合的控制参数值.

3.3　ESADE 算法的自适应机制分析

为了分析 ESADE 算法的自适应机制, 本文使用 50 维标准测试函数的实验结果绘制了图 3 和图 4.

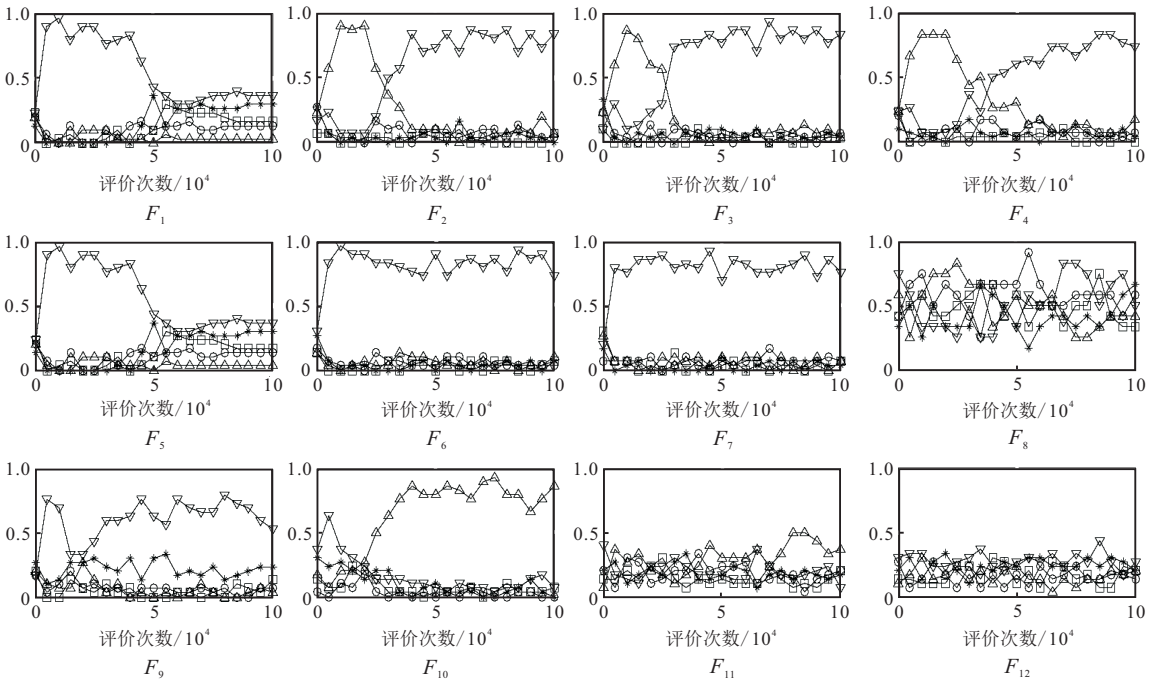

图 3　50 维标准测试函数运行 30 次的平均策略自适应曲线

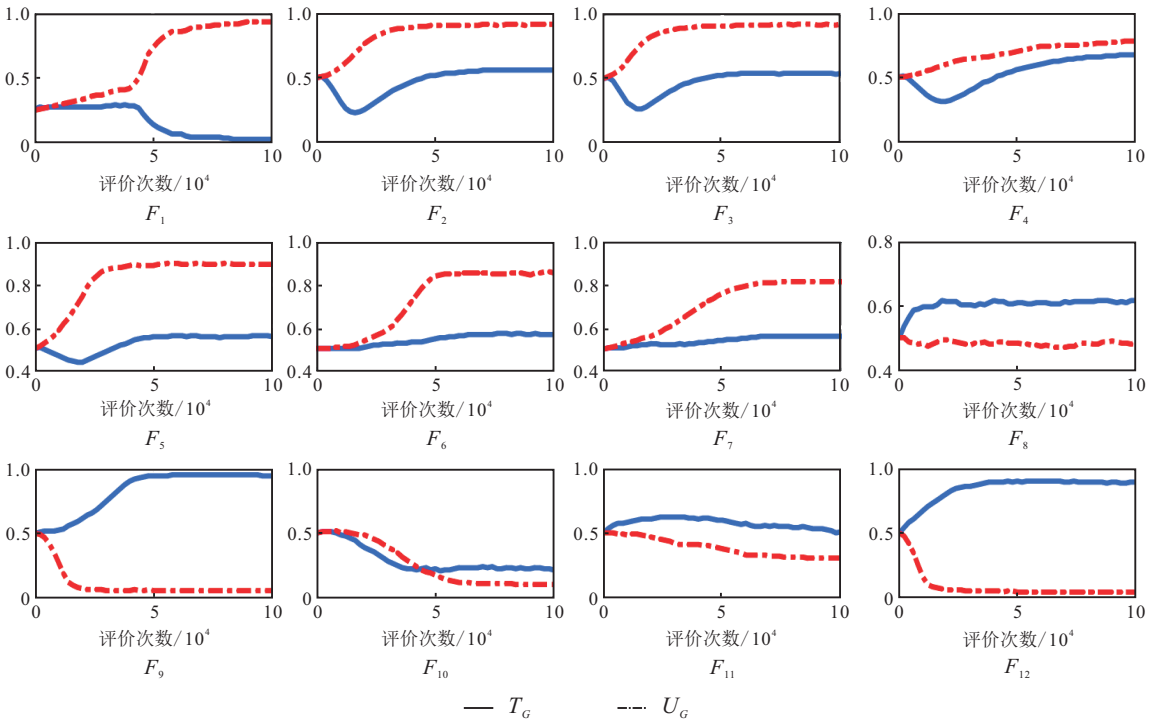

图 4　50 维标准测试函数运行 30 次的自适应参数平均进化曲线

图3为运行30次的平均策略自适应曲线, 纵轴为每个策略运行30次的平均使用百分比, 横轴为函数评价次数. 显然, ESADE算法会根据寻优问题和进化阶段的不同以及历史经验自动选择适合的变异策略. 如果某个策略在先前的进化中产生更好个体的概率高, 那么在以后的进化中其被选择的可能性就大, 也就意味着其使用百分比高. 当其产生更好个体的概率降低时, 就将导致其被选择的可能性减小, 该策略的使用百分比也会随之下降.

图4为自适应参数T_G和U_G运行30次的平均进化曲线, 纵轴为自适应参数T_G和U_G运行30次的均值, 横轴为函数评价次数. 图4很明显地表现出了随着算法的不断进化, ESADE算法会根据历史经验不断地更新自适应参数T_G和U_G来满足不同进化阶段的要求, 从而实现了变异比例因子F和交叉率CR的自适应. 显然, ESADE算法控制参数自适应机制的效果要比随机产生控制参数值和只根据进化代数产生控制参数值的方法要好得多, 因为后两者没有考虑寻优问题的不同和产生个体质量的优劣.

4 结 论

本文提出一种新型自适应差分进化算法——ESADE算法. 策略和控制参数自适应的引入, 使ESADE算法在面对不同的寻优问题和不同的进化阶段时都表现出了优异的性能. ESADE算法的策略自适应机制是根据每个变异策略产生更好个体的概率, 使用指数平滑法预测下一代策略选择概率, 并使用轮盘赌选择法为每个个体选择变异策略. 而控制参数自适应机制是根据产生更好个体的控制参数值, 使用指数平滑法预测下一代自适应参数, 并根据新的自适应参数产生正态分布随机数和柯西分布随机数作为下一代控制参数值. 最后, 本文使用标准测试函数对ESADE算法与其他差分进化算法进行仿真对比, 实验结果表明, ESADE算法具有收敛速度快、收敛精度高等特点, 其性能明显优于其他对比算法.

参考文献(References)

[1] Storn R, Price K. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[2] Rogalsky T, Kocabiyik S, Derksen R. Differential evolution in aerodynamic optimization[J]. Canadian Aeronautics and Space J, 2000, 46(4): 183-190.

[3] Ilonen J, Kamarainen J K, Lampinen J. Differential evolution training algorithm for feeFig1d-forward neural networks[J]. Neural Process Letters, 2003, 17(1): 93-105.

[4] Storn R. Differential evolution design of an IIR-filter[C]. Proc of IEEE Int Conf on Evolutionary Computation. Nagoya: IEEE, 1996: 268-273.

[5] Chen Y, Mahalec V, Chen Y, et al. Reconfiguration of satellite orbit for cooperative observation using variable-size multi-objective differential evolution[J]. European J of Operational Research, 2015, 242(1): 10-20.

[6] Brest J, Greiner S, Boskovic B, et al. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems[J]. IEEE Trans on Evolutionary Computation, 2006, 10(6): 646-657.

[7] Zou D, Wu J, Gao L, et al. A modified differential evolution algorithm for unconstrained optimization problems[J]. Neurocomputing, 2011, 120(10): 1608-1623.

[8] Pan Q K, Suganthan P N, Wang L, et al. A differential evolution algorithm with self-adapting strategy and control parameters[J]. Computers and Operations Research, 2011, 38(1): 394-408.

[9] Mallipeddi R, Lee M. An evolving surrogate model-based differential evolution algorithm[J]. Applied Soft Computing, 2015, 34(1): 770-787.

[10] Fan Q, Yan X. Self-adaptive differential evolution algorithm with discrete mutation control parameters[J]. Expert Systems with Applications, 2015, 42(3): 1551-1572.

[11] Suganthan P N, Hansen N, Liang J J, et al. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization[R]. Singapore: Nanyang Technological University, 2005.

[12] Baatar N, Zhang D, Koh C S. An improved differential evolution algorithm adopting-best mutation strategy for global optimization of electromagnetic devices[J]. IEEE Trans on Magnetics, 2013, 49(5): 2097-2100.

[13] Gong W, Cai Z. Differential evolution with ranking-based mutation operators[J]. IEEE Trans on Cybernetics, 2013, 43(6): 2066-2081.

[14] Asafuddoula M, Ray T, Sarker R. An adaptive hybrid differential evolution algorithm for single objective optimization[J]. Applied Mathematics and Computation, 2014, 231(1): 601-618.

[15] Storn R, Price K. Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces[J]. J of Global Optimization, 1997, 11(4): 341-359.

[16] Mezura-Montes E, Velázquez-Reyes J, Coello Coello C A. A comparative study of differential evolution variants for global optimization[C]. Proc of the 8th Annual Conf on Genetic and Evolutionary Computation. Mexico: ACM, 2006: 485-492.

(责任编辑: 齐 霁)