计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(6):209−212 [doi: 10.15888/j.cnki.csa.006912]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于搜索空间大小的动态变异算子差分进化算法①

苗晓锋[1], 刘志伟[2]

[1](榆林职业技术学院神木校区 信息中心, 神木 719300)
[2](西北工业大学 信息中心, 西安 710072)
通讯作者: 苗晓锋, E-mail: 89019870@qq.com

摘　要: 差分进化算法 (DE) 是一种较新的进化计算技术, 具有概念简单、易于实现、收敛速度快等优点, 得到了广泛的关注和应用. 为了解决经典 DE 计算开销大, 参数设置与问题本身过于相关等缺陷, 提出了一种改进的差分进化算法 (IDE), 它采用了一种动态变异算子, 可根据进化代数的增加, 基于搜索空间大小, 实时地调整变异步长, 从而提高算法的求解精度. 通过在 MATLAB 仿真环境下对著名的基准测试函数分别进行求解, 将改进后的算法和已有的多种优化算法进行比较, 结果表明, 改进的 IDE 算法性能明显优于已知的算法, 证明动态变异是一种有效的改进思路.

关键词: 遗传算法; 优化算法; 动态变异; 差分进化; 仿真; MATLAB

引用格式: 苗晓锋,刘志伟.基于搜索空间大小的动态变异算子差分进化算法.计算机系统应用,2019,28(6):209–212. http://www.c-s-a.org.cn/1003-3254/6912.html

Differential Evolution with Dynamic Mutation Based on Search Space

MIAO Xiao-Feng[1], LIU Zhi-Wei[2]

[1](Information Center, Yulin Vocational and Technical College at Shenmu, Shenmu 719300, China)
[2](Information Center, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: Differential Evolution (DE) is a novel evolutionary computation technique, which has attracted much attention and wide applications for its simple concept, easy implementation and quick convergence. In order to tackle much overhead, problem-dependent parameters, etc and enhance the precision of classical DE, an Improved DE(IDE) algorithm is proposed by using an dynamical mutation operator adjusting the step size based on search space with evolution. Experiments of solving well-known benchmark functions in MATLAB show the improved approach outperforms existing algorithms, and dynamic mutation is a effective improvement ideas.

Key words: genetic algorithm; optimization; dynamic mutation; differential evolution (DE); simulation; MATLAB

差分进化 (Differental Evolution, DE) 是由 Price 和 Storn[1]首先提出的一种简单高效的基于群体的随机搜索算法. 作为遗传算法的一个分支, DE 的性能在很大程度上受到其控制参数 (收缩因子和杂交概率) 的影响[2,3]. 选择不同的策略和控制参数会导致不同的算法性能, 尤其会在很大程度上决定最终所能获得最优解的求解效率和质量[4]. 而选择适当的参数值是一个与待求解问题自身特征相关的问题, 通常由使用者

① 基金项目: 国家自然科学基金 (61672433); 榆林职业技术学院神木校区 2018 年校级教科研课题重点项目 (ZK-201801)
Foundation item: National Natural Science Foundation of China (61672433); Year 2018, Major Educational and Scientific Research Project of Yulin Vocational and Technical College at Shenmu (ZK-201801)
收稿时间: 2018-12-05; 修改时间: 2018-12-25; 采用时间: 2018-12-28; csa 在线出版时间: 2018-05-25

根据主观经验决定. 许多研究都尝试更好地解决这个问题, 例如自适应控制参数 DE(SADE)[4], 模糊 DE(FADE)[5], 自适应 DE(SaDE)[6]和相邻搜索自适应 DE(SaNSDE)[7]等改进 DE 算法.

本文引入一种可根据进化代数实时调整变异策略步长的动态变异算子, 以变异策略的改变来优化 DE 性能, 并将采用了这种动态变异算子的 IDE (Improved DE) 算法和已有的 SADE[4], 古典进化规划 (CEP)[8]和快速进化规划 (FEP)[8]等既有算法进行了仿真比较研究.

1 差分进化算法

差分进化 (DE) 算法是一种基于种群和定向搜索策略的遗传算法[1]. 它的求解过程与其它仿生算法类似, 都是从一个随机生成的初始解种群开始, 模仿生物进化过程中基因变异和杂交的过程进行迭代求解, 直至找到最终最优解.

DE 有多种最基本的形式[1], 最著名的一种是标准 DE 即“DE/rand/1/bin”. 该算法工作时首先随机生成一组初始解, 然后通过变异和杂交操作产生一组对应的试验解, 再根据最优适应值函数判断哪些试验解能作为下一代解种群成员, 然后迭代进行以上操作, 直至所得到当前解种群的精度达到要求时即停止求解循环.

首先, 定义$X_{ri,G}(i=1,2,\cdots,N_p)$是第 G 代种群的解向量, 其中 N_P 是种群的大小,

变异操作: 每个 G 代种群中的解 $X_{ri,G}$ 变异生成一个试验解 $V_{i,G}$, 定义如下

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

其中, $i=1,2,\cdots,N_p$, r_1, r_2 和 r_3 是集合$\{1,2,\cdots,N_p\}$中任意互不相等随机整数, F 是缩放系数.

杂交操作: 如同其它遗传算法一样, DE 也利用杂交算子结合两个不同的解来生成试验解, 该试验解定义如下:

$$U_{i,G}=(U_{1i,G},U_{2i,G},\cdots,U_{Di,G},)$$

其中, $j=1,2,\cdots,D$(D 是问题维数).

$$U_{ji,G}=\begin{cases}V_{ji,G}, & \text{if } rand_j(0,1)\le CR \vee j=k\\ X_{ji,G}, & \text{otherwise}\end{cases} \tag{2}$$

其中, CR 是预先定义好的杂交概率, $rand_j(0,1)$是 (0, 1) 范围内的任意随机数, $k\in\{1,2,\cdots,D\}$也是随机数.

选择操作: 决定 $U_{i,G}$ 和 $X_{i,G}$ 二者当中哪一个成为下一代 G+1 种群的成员. 对求最优值问题而言, 能得到更好目标值的解将被选中继续进行迭代运算.

2 根据搜索空间大小动态调整变异算子的算法

经典 DE 算法的性能很大程度上受到其控制参数 (收缩因子和杂交概率) 的影响, 不同的参数选择和变异策略会导致不同的算法性能.

在已有研究成果中, Yao 和 Liu[9]提出了一种引入三角变异算子的 DE, 可以在算法的收敛速度和鲁棒性两者间取得较好的平衡. He 等[10]采用了辅助种群和放大因子 F 的自动计算. Shi[11]提出了结合 DE 和分布估计的 DE/EDA 算法. Liu 和 Lampinen[3]提出了一种模糊自适应 DE(FADE), 它使用一个模糊逻辑控制器设置杂交与变异概率. Brest 等[5]研究了采用自适应控制参数的 DE(SADE). SADE 采用自适应控制机制调整参数 F 和 CR. Qin 和 Suganthan[4]提出了一种自适应 DE(SaDE), 研究参数 CR 和变异策略的适应性. Yang 等人[6]提出了邻域搜索策略 DE(NSDE), 它利用服从高斯和柯西分布的随机数生成参数 F, 而不是预定义常数 F. 基于 SaDE 和 NSDE, Yang 等人[6]提出了另一个版本的 DE(SaNSDE), 它吸收了 SaDE 和 NSDE 的优点. 苗晓锋等[12]提出了一种基于混合策略的 HDE. 虽然以上方法都采用加权因子的方法来控制变异步长, 但很难解决待求解问题被自身特征支配的缺陷.

本文提出了一种新的动态变异算子, 即根据当前搜索空间的大小动态调整局部搜索的步长, 算子定义如下:

$$x_j(t+1)=x_j(t)+[b_j(t)-a_j(t)]*rand() \tag{3}$$

$$a_j(t)=\mathrm{Min}\left(x_{ij}(t)\right),\ b_j(t)=\mathrm{Max}\left(x_{ij}(t)\right) \tag{4}$$

$$i=1,2,\cdots,ps,\ \ j=1,2,\cdots,n \tag{5}$$

其中, x_j 是解个体 x 的第 j 维向量, $a_j(t)$ 和 $b_j(t)$ 分别是当前搜索空间第 j 维的最小值和最大值. rand() 是[0, 1]区间上的随机数, PS 是种群规模大小, t = 1, 2, …, 指进化代数.

采用了该算子的 IDE 算法流程如下:

Begin
While $NE < MAX_{NE}$ do
For i = 1 to PS do
根据式 (1) 和式 (2) 生成一个试验向量;
计算试验向量的适应值;
在 X_i 和试验向量中选择具有更好适应度的一个进入下一代种群;

End For
根据式 (4) 计算 $a_j(t)$ 和 $b_j(t)$;
根据式 (3) 对最优个体 best 进行变异操作;
if best(t+1) 的适应值优于 best 的适应值
best = best(t+1);
if end
End While
End

其中, PS 是种群规模, best 是当前最优解, NE 是函数运行次数, MAX_{NE} 是函数运行最大次数.

在 IDE 算法中, $b_j(t)–a_j(t)$ 可以被视为当前种群搜索空间变异步长的放大率. 在进化初期, 初始搜索空间大, 步长 $b_j(t)–a_j(t)$ 也大, 此时较大的步长有利于覆盖全局搜索, 便于发现潜在的更优解, 同时加快算法收敛. 随着代数的增加, 种群将逐渐收敛到当前最优值附近, 此时当前种群的搜索空间和步长 $b_j(t)–a_j(t)$ 均变小. 在这种情况下, 较小的 $b_j(t)–a_j(t)$ 值将更有利于在小范围局部搜索.

通过对著名的基准测试函数 simple Sphere's problem 进行求解, 并选取当前解第一维 $b(t)–a(t)$ 的变化进行记录, 结果如图 1 所示.

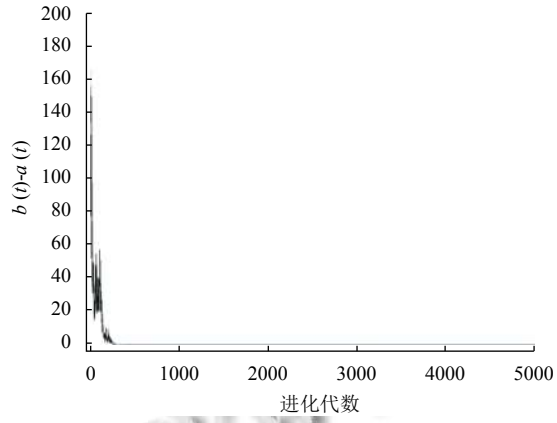

图 1 $b(t)–a(t)$ 值的收敛过程

可见, 在进化开始时 $b(t)–a(t)$ 值和最优适应值很大, 在进化的最后阶段, 最优适应值和 $b(t)–a(t)$ 很小, 算法迅速收敛.

3 基准函数测试试验

为了测试算法性能, 选择了三个单峰函数 (f_1-f_3) 和一个多峰函数 (f_4) 来进行 MATLAB 环境下的仿真求解实验. 表 1 中给出了所有基准测试函数、变量维数、定义域及其全局最优解.

表 1 测试函数

测试函数	n	X	$f_{\min}$
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	[–100, 100]	0
$f_2(x)=\sum_{i=}^{n} \lvert x_i\rvert+\prod_{i=1}^{n} x_i$	30	[–1.28, 1.28]	0
$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i} x_j\right)^2$	30	[–5.12, 5.12]	0
$f_4=-20*\exp\left(-0.2*\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+e$	30	[–600, 600]	0

我们将常用的 CEP、FEP、SADE 和 IDE 四种算法进行比较, 四种算法分别对四个测试函数进行求解运算. 参数设置如下:

种群规模 PS 设置为 50, 控制参数 CR 和 F 分别设置为 0.9 和 0.5, 函数调用最大次数 MAX_{NE} 分别设置为 150 000 (f_1 和 f_4), 200 000 (f_2), 500 000 (f_3).

表 2 中给出了测试函数的求解结果, 其中 Mean 表示上一代最优解的平均值, STD 代表标准方差.

显然, 在同等条件下, SADE 对于前三个单峰函数的求解精度平均优于 CEP 和 FEP 多达 e+24 倍、e+20 倍和 e+12 倍, 对第四个多峰函数的求解精度平均优于两种算法 e+15 和 e+13 倍. 而 IDE 的求解精度又远优于 SADE, 上述优势分别达到了 e+27、e+25 和 e+16 倍. 只是在最后一个多峰函数上, IDE 和 SADE 的求解精度数量级相同, 但它的当前最优解的值比 SADE 的值更接近于最终全局最优解.

其中 IDE 对测试函数 f_1 和 f_3 求解时的收敛过程分别如图 2 和图 3 所示, 可见其收敛过程也非常迅速, 并未陷入局部最优.

表 2 测试结果

测试函数	CEP Mean (Std Dev)	FEP Mean (Std Dev)	SADE Mean (Std Dev)	IDE Mean (Std Dev)
f_1	2.2e–04(5.9e–04)	5.7e–04(1.30e–04)	1.1e–28(1.0e–28)	8.24e–55(6.52e–55)
f_2	2.6e–03(1.7e–04)	8.1e–03(7.70e–04)	1.0e–23(9.7e–24)	1.26e–48(3.45e–49)
f_3	5.0e–02(6.6e–02)	1.6e–02(1.40e–02)	3.1e–14(5.9e–14)	4.63e–30(5.87e–30)
f_4	9.2(2.8)	1.8e–02(2.10e–03)	7.7e–15(1.4e–15)	4.14e–15(5.61e–15)

4 结论与展望

本文提出了一种改进的 IDE 算法, 该方法可以根据当前搜索空间的大小动态调整变异步长. 四个著名的基准测试函数求解实验表明, 所提出的 IDE 算法在同等条件下求解精度大大优于 CEP、FEP 和 SADE. 今后的工作是尝试更多的进化和参数设置策略, 以研究其对 DE 算法性能的影响.

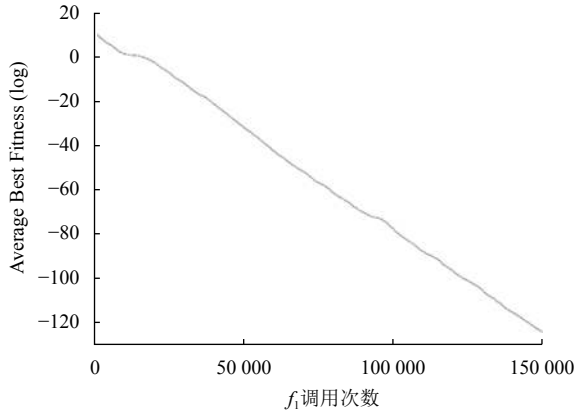

图 2 改进 DE 求解 f_1 时的收敛过程

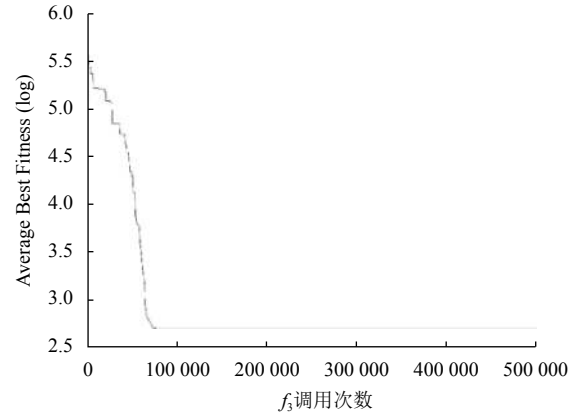

图 3 改进 DE 求解 f_3 时的收敛过程

参考文献

1 Storn R, Price K. Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341–359. [doi: 10.1023/A:1008202821328]

2 Vesterstrom J, Thomsen R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. Proceedings of the 2004 Congress on Evolutionary Computation. Portland, OR, USA. 2014. 1980–1987.

3 Liu JH, Lampinen J. A fuzzy adaptive differential evolution algorithm. Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2005, 9(6): 448–462.

4 Qin AK, Suganthan PN. Self-adaptive differential evolution algorithm for numerical optimization. Proceedings of 2005 IEEE Congress on Evolutionary Computation. Edinburgh, Scotland, UK. 2005. 1785–1791.

5 Brest J, Greiner S, Boskovic B, *et al*. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation, 2006, 10(6): 646–657. [doi: 10.1109/TEVC.2006.872133]

6 Yang ZY, Yao X, He JS. Making a difference to differential evolution. Siarry P, Michalewicz Z. Advances in Metaheuristics for Hard Optimization. Berlin, Heidelberg: Springer, 2018. 397–414.

7 Ali MM, Törn A. Population set-based global optimization algorithms: Some modifications and numerical studies. Computers and Operations Research, 2004, 31(10): 1703–1725. [doi: 10.1016/S0305-0548(03)00116-3]

8 Sun JY, Zhang QF, Tsang EPK. DE/EDA: A new evolutionary algorithm for global optimization. Information Sciences, 2005, 169(3–4): 249–262.

9 Yao X, Liu Y, Lin GM. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation, 1999, 3(2): 82–102. [doi: 10.1109/4235.771163]

10 He YC, Kou YZ, Shen CP. An improved differential evolution based on triple evolutionary strategy. Proceedings of the 3rd International Symposium on Intelligence Computation and Applications. Wuhan, China. 2008. 89–97.

11 Shi Y, Lan ZZ, Feng XH. Differential evolution based on improved learning strategy. Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence. Hanoi, Vietnam. 2008. 880–889.

12 苗晓锋, 刘志伟. 一种基于混合策略的差分进化算法研究. 计算机应用与软件, 2019, 36(3): 260–265.