基于IFEP的自适应下界改进方法

吴国桐  
计算机科学与工程系深圳，中国  
wugt2022@mail.sustech.edu.cn

*摘要*—在传统遗传规划算法（EP）中，采用自适应步长，在搜索初期有着较快的收敛速度，而在后期可以通过减小步长逐渐逼近全局最优。本文首先对传统EP算法进行了深入研究和复现，并在此基础上，动态调整步长下界，并且改进了的初始值设定方法。由于柯西分布在距离最优解距离较远时效果更好，反之高斯分布在距离较近时更好，因此本文在改进EP方法（IFEP）基础上，使用两个独立互不干扰的种群，根据一种新的种群选择方法，可以更好的结合两者优点。实验表明，本文提出的改进方法相对于原有的EP算法有着更快的收敛速度和更强的搜索的能力。

关键词—遗传算法，自适应步长，种群选择方法

# 引言

使用EP求解最优化问题，个体通常采用实数型编码，在没有先验知识的前提下，目的是寻找全局的最优解。与普通遗传算法相比，EP只会采用变异的方法来产生新解，并采用自适应步长的调整策略——在搜索开始的时候，步长较大从而收敛更快；而随着搜索的进行，自适应减小步长，从而最终能收敛于全局最优。

在变异过程中，经典EP方法（CEP）使用高斯分布来产生新的个体。由于高斯分布在远离均值处概率密度极小，因此在搜索之初距离最优解较远时，高斯分布产生的新解收敛非常慢。另外一种基于柯西分布的快速EP方法（FEP）仅仅在变异时采用柯西分布，其它与CEP保持不变，实验结果证明，柯西分布在搜索之初能产生距离父辈较远的子代，从而收敛速度加快；然而对于一些初始解与最优解距离较近的测试问题上，FEP分布反而效果更差。基于以上两种分布的特点，提出了一种改进的EP方法（IFEP），每个父代分别通过高斯分布和柯西分布变异产生两个子代，然后在锦标赛选择环节上，所有父代和子代都参与选择，选出每次迭代中胜利次数最多的个体作为新一代的种群，其它与FEP和CEP保持不变。实验结果证明，IFEP能够结合两者的优点，有着更快的收敛速度和更优的搜索结果。

EP方法最大的特点，即在于动态调整步长，而决定步长的关键变量，同样作为个体中的一个基因，在搜索过程中不断演化。在论文中[1]，对于所有的测试问题，的初始值被设置为一个常量3，这样必然带来很低的泛用性。可以发现随着演化代数的增加，的值逐渐减小，大约在10000代后， 的值已经非常趋近于0。如图1所示，当值非常小时，搜索步长几乎为0，效率大为降低。一种改进方法是对设置一个下界，从而避免让的值趋向于0，然而在论文[1]中，整个搜索过程这个下界始终固定不变，而不能根据具体的问题和搜索过程进行调整，同样会带来较差的泛用性。因此本文的主要工作是对的初始化和下界的动态调整做出了优化，以期改进上述问题。

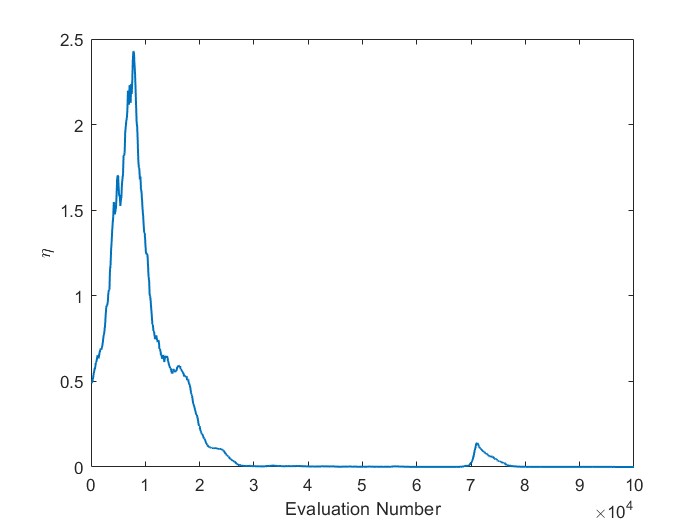


图1. 使用CEP求解ackley问题，随着演化的进行，的值逐渐趋于0。其中横坐标表示演化中评估的次数，纵坐标为的值的变化。

虽然实验证明IFEP能够结合两种分布的优点，取得了较CEP和FEP更好的结果，但是IFEP的结合方法是线性的，即每个个体需要同时产生两种分布的子代，然后将子代和父代简单聚合在一起。本文使用两个独立的种群，分别产生高斯分布和柯西分布的个体，然后交叉进行锦标赛选择，而不是使用IFEP的简单聚合过程，具体细节将在算法部分进行介绍。

# 算法框架

本文在IFEP的基础上，采用基于自适应下界调整的EP方法（AFEP），主要做出了以下改进工作：

1. 改进了的初始值设定，根据决策变量的范围和维度来定义的初始值，使其可以更好适应不同测试问题。
2. 改进了的下界设定，文献[2]建议使用下一代种群的中位数来作为其动态下界。但是本文在具体实践过程中，发现该方法依赖于调整的频率，如果每一代都进行下界调整，最后的下界反而会呈现上升趋势，导致搜索失败。因此本文将的下界同样作为个体中的一个基因，参照的变异方法进行逐代演化。
3. 使用两个种群来分别表示高斯分布和柯西分布，其变异产生新个体的过程互不干扰。但是在锦标赛选择过程中，为了保持种群的多样性，每个种群的父代个体将同新个体竞争。

具体的算法步骤如下：

1. 初始化：产生个个体作为第一代种群，并设定当前代数。其中高斯分布的种群，以及柯西分布的种群 . 其中为决策变量， 为自适应步长，为的下界。对于的初始化，在其约束条件内进行均匀采样；而对于的初始化方法，根据不同问题的不同决策空间，采用如的初始化方法，其中，为决策变量的维度。根据文献[3]的建议，此处 取值为；而的初始化则直接取 初始值的平均值。
2. 变异：对于每一个个体和，根据自己种群对应的变异算子，分别产生一个新的个体，分别可以用和表示。其中，为表示均值为0，方差为1的高斯分布随机数，且对于每一维决策变量都不相同。对于步长的变异方法，不论高斯分布还是柯西分布的种群，本文都与FEP和CEP保持一致，如式表示。而对于下界，则是参照的变异方法，如式进行演化。
3. 计算适应度函数：根据测试问题函数，分别计算每一个个体和的适应度大小。
4. 选择：对于不同的种群，组成两个不同的联合种群Pop1和Pop2，并分别对每一个联合种群进行锦标赛选择，两个联合种群可以表示为和. 其中锦标赛选择的方法是，将所有个体混合在一起，对于每一个个体，随机抽取个其它的个体，比较和其它个个体的适应度值，并且记下总共获胜的次数
5. 产生下一代种群：对于不同的两个种群Pop1和Pop2，分别选出获胜个体最多的个个体作为下一代种群的父辈。
6. 如果满足算法终止条件则退出循环，否则返回第2步

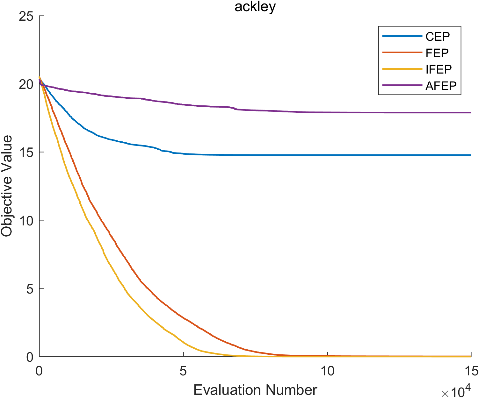
# 实验分析

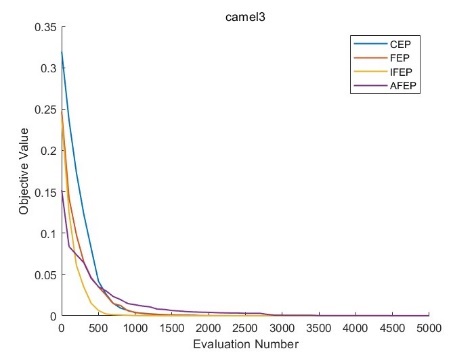
本文使用课程提供的十个单目标测试问题，并在此基础上进行实验。为了进一步理解实验，本文将决策变量的维数取为3，分别给出不同测试问题在三维下的直观图像，如图2展示。可以观察到，其中是sphere函数和step函数，形状为简单的单峰函数，最优值都是在决策空间中心取得。rosenbrock函数和camel3函数形状相似，在决策空间边际变化较大，最终呈现出筒状。而对于noisyQuartic函数、schwefel226函数、ackley函数、holder函数和michal函数都为典型的多模函数，对于同一个目标值，有多个决策空间的点与之对应，因此存在许多局部最优解，将会对最终寻找全局最优解造成较大的困难。

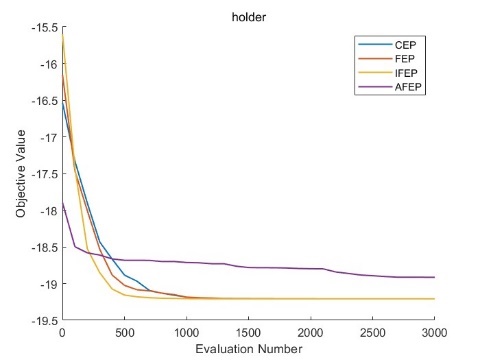
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

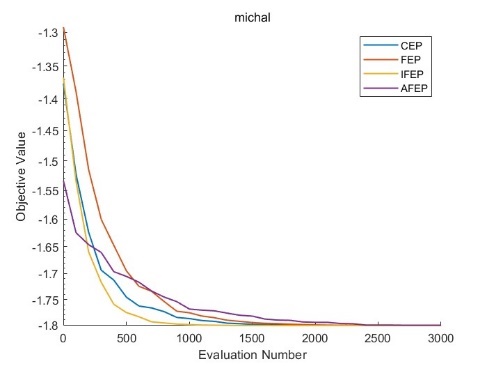
图2. 三维角度下，不同测试问题的目标值形状

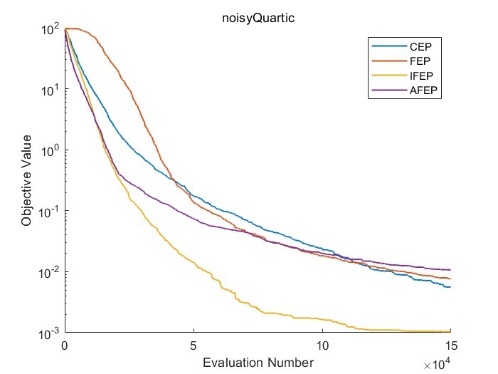
实验测试平台的CPU为AMD Ryzen 5 5600 6-Core Processor，在MATLAB 2022b上进行编程，每个测试问题运行30次求平均值。设定实验的超参数，种群中的个体数量为100，下界初始值为自适应步长初始值的平均值。设定每个问题最大的评估次数为300000，分别实现了CEP、FEP、LFEP和AFEP，图3展示了所有运行过程，不同算法在同一测试问题的平均收敛速度。在shwefel222和shwefel222函数上，AFEP有着最快的收敛速度，并且取得了最好的结果。而在camel3和michal函数上，AFEP的收敛速度普遍比FEP和CEP更快，而不及IFEP方法，然而取得的最优值和三种方法相差无几。

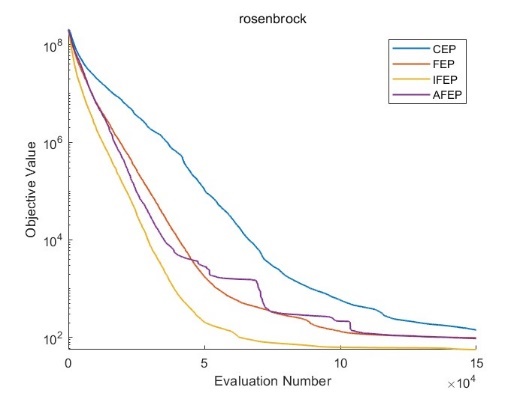


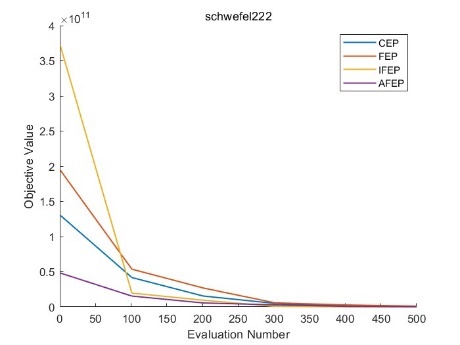


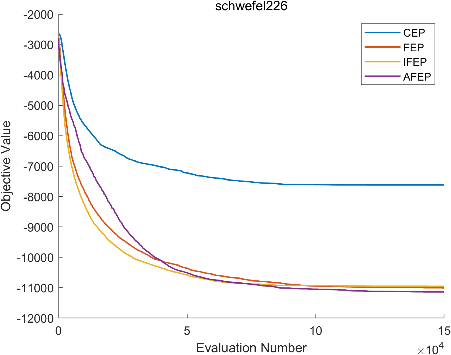


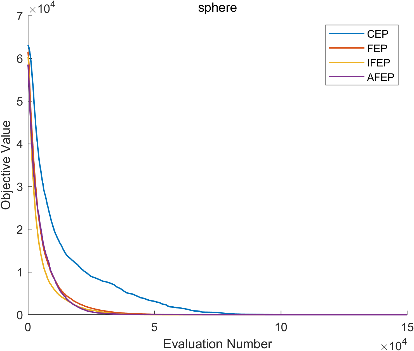












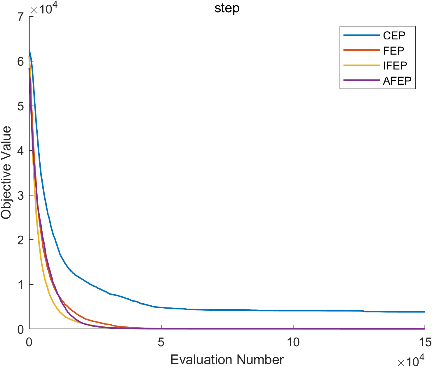


图3. 对于同一测试问题，CEP、FEP、LFEP、AFEP算法的平均收敛速度。注意此处为了更直观的观测，在一些测试问题的y轴取对数坐标表示，并且限制了一些问题的最大评估次数。

为了仔细探究本文对于初始值方法的改动作用，从图3可以发现在演化进行之初，AFEP的目标值较其它方法都更优，并在所有测试问题上都有如此现象。为了更好观测，以ackley测试问题为例，取前500评估次数绘制图像，如图4可以说明该初始值方法在演化之初可以取得更好的效果。

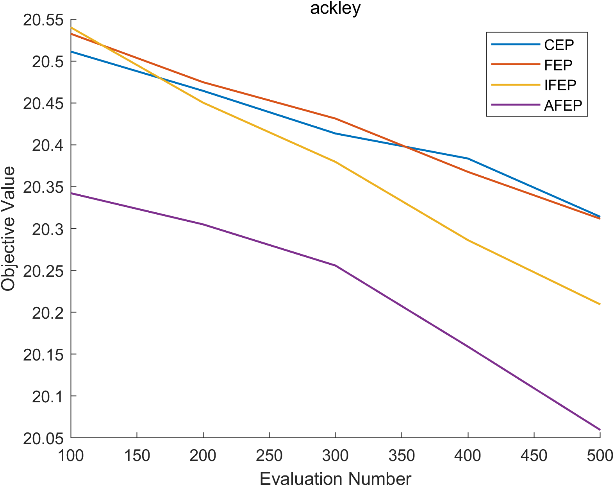


图4. 前500评估次数中，在ackley测试函数上目标值的变化趋势

然而在ackley测试函数上，AFEP的表现则相较于其它方法有着最差的表现。为了深入探究其原因，由于本文将下界作为一个种群个体参与演化，图5表现了整个演化过程中平均值的变化。可以发现在演化之初，的值较小，与FEP和CEP过程中的的变化相差无几。但是在演化的后期，值突然急剧变大。一种可能的解释是，由于演化后期非常接近最优目标，每一代的取得的进步非常小， 为了跳出局部最优，向增大的趋势进行演化，但是这种趋势在ackley和holder问题上并不适用，因此取得的效果更差。

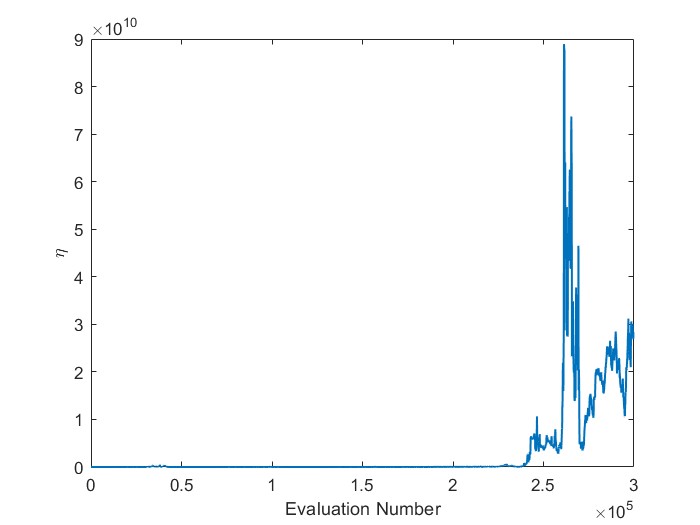


图5. 使用AFEP求解ackley问题，随着演化的进行，值的变化。其中横坐标表示演化中评估的次数，纵坐标是的值的变化。

而对于本文采用了两个互不干扰的种群，来改进原有LFEP对于两种分布的组合方式。因此本文采用The Wilcoxon signed ranks test来衡量AFEP相对于其它方法的表现，具体数据如表1所示。可以发现虽然在ackley问题上效果不佳，但在同样的评估次数上，普遍取得了相较于其它CEP、IFEP更优秀的结果，而最终的结果和FEP相近，证明本文所采用的两种分布的组合方式是有助于寻优的。

# 结论

本文在LFEP的基础上，提出了AFEP，主要针对自适应步长的初始化和下界进行了动态设置，同时使用两个互不干扰的种群，改进了原有的LFEP对于高斯分布和柯西分布的线性组合方式。实验结果表示，AFEP结合了CEP和FEP各自的优点，而且相对于LFEP更优秀的搜索能力。

然而本文将的下界作为种群中的一个个体进行演化，虽然在一些问题上取得了较好的结果，但是在一些特殊的测试问题上表现不佳，未来应该在此基础上做进一步的优化。另外，目前本文的工作主要采用实验的方法进行验证，而在理论上缺乏进一步的深入研究，这也可以作为未来工作方向之一。

##### References

1. Xin Yao, Yong Liu and Guangming Lin, "Evolutionary programming made faster," in IEEE Transactions on Evolutionary Computation, vol. 3, no. 2, pp. 82-102, July 1999, doi: 10.1109/4235.771163.
2. Liang, Ko-Hsin, Xin Yao, and Charles S. Newton. "Adapting self-adaptive parameters in evolutionary algorithms." Applied Intelligence 15 (2001): 171-180.
3. Mallipeddi, Rammohan, S. Mallipeddi, and Ponnuthurai N. Suganthan. "Ensemble strategies with adaptive evolutionary programming." Information Sciences 180.9 (2010): 1571-1581.

**tuyour paper may result in your paper not being publi**

表1. 不同测试问题上，对于不同算法的Wilcoxon Signed Ranks Test的计算结果。其中表中每一个单元格的第一行数据为30次运行的平均值，第二行数据为30次运行的方差，加粗为该测试问题中表现最佳的算法

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Problem** | **N** | **M** | **D** | **FE** | **CEP** | **IFEP** | **FEP** | **AFEP** |
| rosenbrock | 100 | 1 | 30 | 300000 | 6.8326e+1 (5.67e+1) = | 5.8048e+1 (4.08e+1) = | **4.2198e+1 (3.22e+1) =** | 6.0249e+1 (5.10e+1) |
| camel3 | 100 | 1 | 2 | 300000 | 1.7896e-11 (2.23e-11) - | 1.9559e-11 (2.29e-11) - | 2.4920e-11 (2.35e-11) - | **0.0000e+0 (0.00e+0)** |
| holder | 100 | 1 | 2 | 300000 | -1.9209e+1 (2.20e-10) - | **-1.9209e+1 (1.67e-10) -** | -1.9209e+1 (2.06e-10) - | -1.8709e+1 (7.14e-15) |
| schwefel222 | 100 | 1 | 30 | 300000 | -2.0268e+2 (2.02e+1) - | -2.2579e+2 (3.73e+0) - | -2.2700e+2 (2.90e+0) - | **-2.4779e+2 (1.02e+1)** |
| schwefel226 | 100 | 1 | 30 | 300000 | -7.6633e+3 (6.88e+2) - | -1.0983e+4 (3.26e+2) = | -1.1060e+4 (3.59e+2) = | **-1.1122e+4 (4.55e+2)** |
| ackley | 100 | 1 | 30 | 300000 | 1.6226e+1 (2.45e+0) + | **4.2151e-3**  **(2.54e-4) +** | 1.0761e-2 (1.06e-3) + | 1.7860e+1 (6.06e+0) |
| michal | 100 | 1 | 2 | 300000 | -1.8013e+0 (2.83e-10) - | -1.8013e+0 (6.27e-10) - | -1.8013e+0 (8.63e-10) - | **-1.8013e+0 (9.03e-16)** |
| noisyQuartic | 100 | 1 | 30 | 300000 | **8.5895e-4**  **(6.65e-4) +** | 1.1592e-3 (1.00e-3) = | 2.5278e-3 (1.18e-3) + | 2.5293e-3 (5.63e-3) |
| sphere | 100 | 1 | 30 | 300000 | **2.5122e-5**  **(3.49e-6) +** | 3.3717e-5 (4.24e-6) + | 2.2185e-4 (3.81e-5) + | 3.7390e-1 (1.79e+0) |
| step | 100 | 1 | 30 | 300000 | 5.4515e+3 (4.77e+3) - | 8.3333e-1 (1.15e+0) = | **0.0000e+0** **(0.00e+0) +** | 1.0433e+1 (4.96e+1) |
| +/-/= | | | | | 3/6/1 | 2/4/4 | 4/4/2 |  |