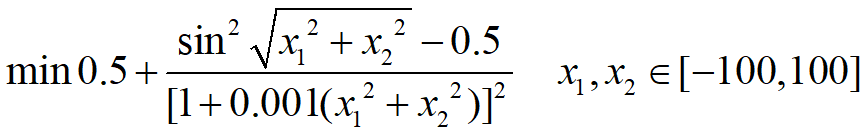
信号处理的智能化方法及应用 第五次作业

1. **基于进化算法或其他智能优化算法编程求解如下最优化问题。**



1. 实现环境：MATLAB R2019a
2. 环境配置：Windows 10 Enterprise Version 2004, AMD Ryzen R7 2700
3. 实现方法：构造了一个函数，负责通过函数指针调用不同的搜索方法实现，并计算每种方法的最小/平均/最大耗时。
4. 运行结果

程序对比运行了5份代码，分别是：首先提供的基础进化优化代码、在网上代码的基础上修改而成的免疫算法代码、后续提供的优化后的进化算法、小生境算法和猴王算法，同时对基础优化代码的不同参数下的运行效果进行了对比，每一个算法运行了100次以避免偶然情况。

1. 基础进化算法

**种群大小为1000，交叉率0.8，变异率0.9，误差限9e-3，计算结果：**

"k: min/avg/max

53 / 692.1 / 1977"

"time: min/avg/max

1.893075 / 24.912441 / 70.977121"

**增加种群大小至2000，其余参数保持不变：**

"k: min/avg/max

49 / 121.4 / 207"

"time: min/avg/max

3.772765 / 9.285405 / 15.759829"

种群大小增加，意味着一次迭代能够搜索、变化的个体增加，搜索力度增大，因此迭代次数和搜索时间均有显著的下降，但是程序占用的内存有一定的提升（因为MATLAB本体也占用内存，因此无法确切指出占用内存的多寡是否与种群大小有直接关系，但我认为应该是有一定的线性关系）。

**减小交叉率至0.5，其余参数保持不变：**

"k: min/avg/max

160 / 544.5 / 1851"

"time: min/avg/max

4.930906 / 16.777650 / 57.063620"

**减小交叉率至0.1，其余参数保持不变：**

"k: min/avg/max

0 / 1340.4 / 2704"

"time: min/avg/max

0.013636 / 31.863193 / 64.200378"

当交叉率过低时，从理论上讲，这会导致产生新的特征的概率降低，增加早熟的风险，因此搜索力度会降低，从实验数据上，搜索时间增加了一些，但是增加幅度不算很大，而迭代次数却显著地增加了，这应该是因为降低了交叉率会导致一个iter的工作量降低，从而单位iter的耗时减少而iter总数增加。对于本机的简单测试用例而言，时间增加不大，但是应该要注意如果求解规模比较大，更多的迭代次数可能需要更大的I/O吞吐量，I/O瓶颈有可能造成更加显著的性能下降、

**减小变异率至0.5，其余参数保持不变：**

"k: min/avg/max

175 / 502.8 / 1272"

"time: min/avg/max

6.342956 / 18.212994 / 46.167151"

**减小变异率至0.1，其余参数保持不变：**

"k: min/avg/max

34 / 144.9 / 526"

"time: min/avg/max

1.197066 / 5.166721 / 18.744069"

比较令人意外的一点是，我原先以为变异有助于新的特性产生，因此变异率高的话应该有助于提升搜索的遍历性，提高搜索速度，但是实际结果却表示，变异率减小反而有助于提升搜索力度。

经过讨论课上老师的解答，我的理解是这样的：在特定的进化问题中，变异率有一个sweet point的区间，并不是越高越好，如果变异率过高，虽然会有更大的概率产生新的特性，但是也造成种群中原有的最佳个体话语权下降，而最佳个体更有搜索潜力，是进化计算算法的核心，变异率过高则最佳个体的特征会被稀释，实质上相当于进化算法部分退化为了传统的随机搜索，因此变异率过高并不能提升搜索力度。

**为了与概率分布优化后的进化算法（第3条）进行对照，种群大小1000，交叉率0.9，变异率0.1，计算结果：**

"k: min/avg/max

21 / 118.3 / 309"

"time: min/avg/max

0.804626 / 4.414780 / 11.477091"

1. 免疫算法

在原先代码的基础上，修改免疫个体为1000个，变异概率为0.9，误差限为9e-3，并调整搜索区间，以方便和提供的进化算法进行对比，其性能如下：

"k: min/avg/max

3 / 85 / 379"

"time: min/avg/max

1.041724 / 15.772979 / 68.839765"

总体性能与进化算法相似。

如果降低变异概率为0.7（原始代码中推荐的变异概率），则性能如下：

"k: min/avg/max

7 / 33.9 / 72"

"time: min/avg/max

1.757468 / 6.595245 / 13.358293"

如果增加免疫个体数量为2000个，性能如下：

"k: min/avg/max

4 / 1.580000e+01 / 54"

"time: min/avg/max

4.452056 / 12.941700 / 40.513180"

与基础进化算法一样，降低变异概率或者提升个体数量都可以有效提升算法的遍历性，从而提升搜素速度，不过在免疫算法中，提升个体数量带来的性能提升没有进化算法显著，收益不大。

1. 优化后的进化算法

与基础的进化算法不同的是，优化算法交叉率增大，变异率减小，同时优化了概率分布：

"k: min/avg/max

2 / 73.3 / 252"

"time: min/avg/max

0.086758 / 2.762733 / 9.415231"

性能相比基础的进化算法在同样的个体数量、变异率和交叉率下又有了一定的提升。从和变异率相似的角度出发理解，概率分布表的调整实质上也是对最优个体的加强，因此可以起到提升搜索性能的效果。

1. 小生境算法

小生境算法的主要目标是提升算法的搜索效率并摆脱早熟，下面是误差限设定为9e-3时的计算性能：

"k: min/avg/max

1 / 20.7 / 40"

"time: min/avg/max

0.258069 / 4.393240 / 8.480705"

性能与优化后的进化算法相当，同时也可以看出，算法在数个iter中就摆脱了0.0097的早熟点。

同时，小生境算法的精度很有限，如果将误差限设定为1e-6：

"k: min/avg/max

100 / 100 / 100"

"time: min/avg/max

20.745324 / 20.947415 / 21.380289"

算法无法在给定的100代以内收敛，如果增加收敛代数限制至10000，可以计算出结果，但是耗时较长，理论来说，像下面的猴王算法一样加入一个局部搜索的机制应该就可以改善精度问题。

1. 改进猴王算法

改进猴王算法是进化算法与局部进化算法的结合，通过局部搜索来增加搜索的精度，同时整体的进化算法通过“猴王”与随机个体的交叉来进行进化计算搜索同时避免早熟。

"k: min/avg/max

10 / 218.9 / 543"

"time: min/avg/max

0.035923 / 0.806553 / 1.985902"

算法的效果非常好，在很短时间内就完成了搜索，同时结果有着很高的精度，误差在1e-8之内。

1. 小结

进化算法属于智能优化的方法，智能优化与传统搜索方法的最大区别就是其搜索过程是有引导的，我认为反映在进化算法上，就是利用在遗传中留下的最优个体包含的信息来指导后续进化的进程。因此进化算法的核心应该是围绕着最优个体进行一些tradeoff，一方面要保证最优个体在进化过程中的话语权，比如变异概率的适当选择能够提升搜索的性能，另一方面也要保证多样性，确保最优个体能够充分接触各个新产生的“可能性”，这也是生物进化中遗传和变异分布对应的左右。基础的猴王算法我觉得其实可以理解为将最优个体完全隔离，仅对其他个体进行充分变异的进化算法，在保护最优个体的同时避免性状单一。

完整代码源文件请查看：

<https://git.nju.edu.cn/Minaduki/Intelligent_Signal_Processing>