智能优化算法及其应用
求解多极小函数

自 66　夏卓凡　2016011496

2019 年 5 月 3 日

1　遗传算法与目标函数

本作业将使用自己实现的简单遗传算法求解一维 Griewank 函数的全局最小值。简单遗传算法的步骤是，初始化种群，按目标函数计算适应度，按适应度进行**选择操作**，之后按照给定概率进行**交叉操作**和**变异操作**，迭代出新一代的种群。在进行一定次数的迭代之后，将种群中的个体解码，得到对应目标函数的自变量的值，代入后得到相应的函数值，即完成了对目标函数的求解，遗传算法的流程图如下所示。

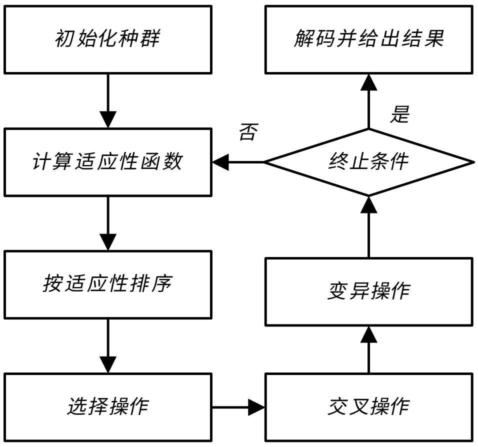

图 1: 遗传算法流程

遗传算法具有不依赖函数本身特性的特点，与传统依赖梯度进行优化的算法不同，遗传算法对局部极小问题有着良好的性能。遗传算法通过交叉和变异这两种产生新解的方法，无视目标函数的限制，这样很容易能跳出一些不好的局部极小值点。此外，遗传算法具有固有的并行性，可以很容易地提高计算速度。但是遗传算法的表示是离散的，在函数优化这一连续问题上会有精度限制，在追求更高精度结果的情况下需要更多的编码长度，这可能对算法的实现造成不利的影响。

Griewank 函数的表达式为

$$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

本作业简化问题，取一维的情形，即

$$f(x)=\frac{x^2}{4000}-\cos x+1 \ x\in[-600,600]$$

下面两图展示了该函数在大尺度和小尺度下的变化特点。

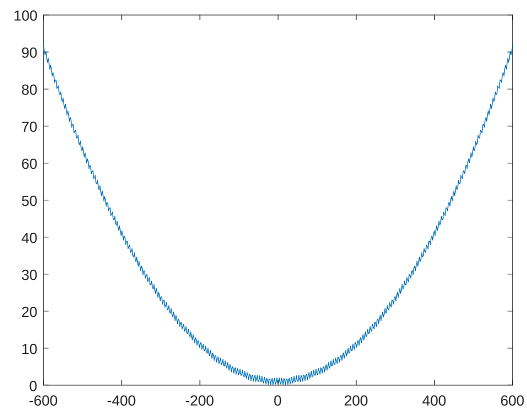

图 2: Griewank 函数 $x\in[-600,600]$ 的图像

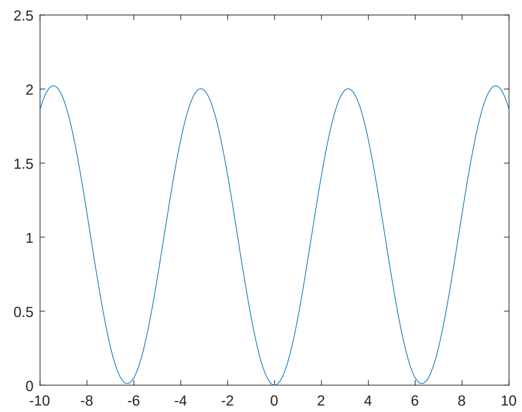

图 3: Griewank 函数 $x\in[-10,10]$ 的图像

从上面两图可以看出，在 $x=0$ 处 $f(x)=0$ 是全局最小值。但是由于此函数在 $x=0$ 附近的区域接近余弦曲线，局部极小值非常多，而且各个极小值之间差异不明显，无法提供有效的优化信息，基于梯度的优化算法很难有效解决这类问题。

2 求解结果分析

首先给出一次迭代求解的目标函数下降曲线，本次结果为 0.000592。从下图可以看出，个体最优值在很早的迭代中就收敛到接近全局最优值的状态，而种群平均值在前几次迭代后下降很快，但在平均值较小之后下降地就比较慢了，之后两个值都一致地趋向全局最优值。

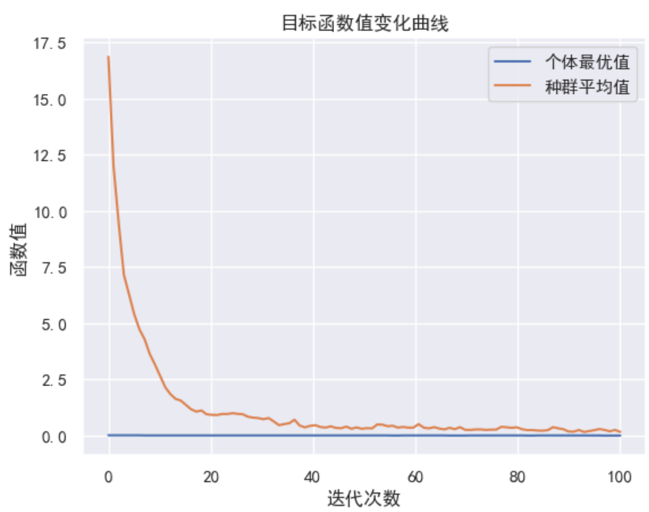

图 4: 一次迭代过程中的目标函数变化曲线

接下来给出 20 次测试的统计结果，最佳性能为 0.000030，最差性能为 0.093829 平均性能为 0.026469，性能标准差为 0.030016。从上面的统计结果可以看出，遗传算法对 Griewank 函数在一维情形下的优化是可以得到令人满意的结果的。

3 实验心得与体会

我在本次实验中手动实现了 SGA 算法，体会到了算法流程中的每一个细节，对遗传算法中的“选择”、“交叉”、“变异”有了深刻的认识，也对优化算法的“评价解”、“产生解”的步骤有了更深的体会。本次作业我选择了比较简单的目标函数，没有在维度上进行探索，这主要是由于自己实现的算法比较基础，还没有考虑高维变量的编码，是一个改进的方向。此外，SGA 算法的超参数，如变异概率、交叉概率、种群数量等，虽然在我选择的问题上影响不是很大，但对于其他问题的意义，还需要进行进一步研究。

本次作业的代码的使用方法如下，各个超参数可以在代码内部设定。

```
python sga.py
```