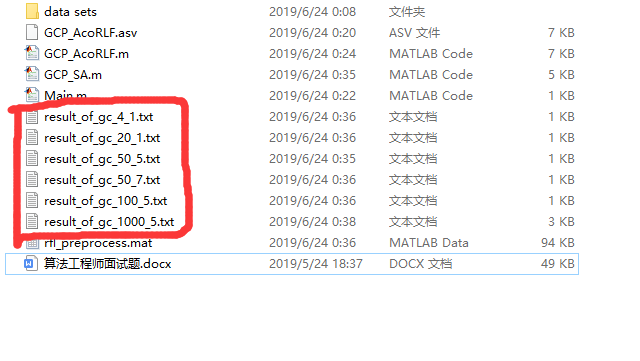
### 程序说明

**使用RLF算法结合智能优化算法（蚁群算法，模拟退火算法）求解**

**解压之后直接运行Main.m文件，计算结果保存在txt文件中。由于算法的随机性，对于某些数据可能要反复运行多次才能得到满足条件的结果**



1. 数据'gc\_4\_1','gc\_20\_1','gc\_50\_7','gc\_100\_5'对应的问题用蚁群+RLF算法求解（ACORLF,对应GCP\_AcoRLF.m）,对应的最优颜色数目分别为2, 3, 14, 17; ACORLF**算法参考了论文：**

**An aco algorithm for the graph coloring problem,2008**

,2，数据'gc\_1000\_5'对应的GCP问题通过RLF方法可直接获得比要求的110更好的结果，这里通过多次随机搜索获得的最佳值为**104**，算法也写在GCP\_AcoRLF.m中

,3，数据’gc\_50\_5’对应的问题通过模拟退火算法求解（GCP\_SA.m），需要反复多次，才能获得颜色数目为9的可行解

### 总结：

1. 在小规模优化问题上，RLF算法比另一种直接局部优化算法DSATUR差，但是在大规模优化问题'gc\_100\_5'上RLF算法似乎更好一些

,2，对于小规模的问题，模拟退火、遗传算法比PSO,差分进化好很多（速度更快，结果更优），其中模拟退火经过多次仿真，可以勉强达到题目要求的结果，遗传算法则要稍差一点。我这里稍微调整了下模拟退火后期的学习率，优化了局部搜索性能。另外，模拟退火的随意扰动环节选择了单点突变算子。

,3，蚁群RLF算法来自参考文献An aco algorithm for the graph coloring problem,2008，其核心思想是：通过蚁群算法的信息素机制，在RLF中选择进入第k种颜色的顶点时，给‘蚂蚁’提供全局信息辅助其进行选择，从而进一步提高RLF的性能。但是实际操作的时候发现，对于大规模的图'gc\_1000\_5'，局部信息（待涂颜色的顶点在所有未涂颜色顶点中的深度）要重要得多，加入全局信息反而恶化了性能，因此对于'gc\_1000\_5'直接用RLF来求解，通过随机初始化的方式进行搜索优化