



华南理工大学  
South China University of Technology

# 《人工智能》课程论文

(2018-2019 学年第 1 学期)

## 求解 TSP 问题的遗传算法设计与实现

学生姓名： 石望华

提交日期： 2018 年 12 月 31 日

学生签名：

学 号	201630676843	座位编号	
学 院	软件学院	专业班级	卓越班
课程名称	人工智能	任课教师	应伟勤
教师评语：			
本论文成绩评定： _____分			

# 求解 TSP 问题的遗传算法设计与实现

石望华

**摘要：**本文针对具体的 TSP 问题，基于 Galib 库论述了遗传算法的设计与实现步骤。主要从四种遗传算法（GASimpleGA、GASteadyStateGA、GAIcrementalGA、GADemeGA）与四类染色体（GAListGenome、GATreeGenome、GAArrayGenome、GABinaryString）的角度进行了类图继承关系研究和对各个方法的作用的详细讨论，并将之作用于四个数据集（berlin52、pr76、rat99、lin105），通过寻找最佳的局部最优解，不断进行调参测试与评估，最终使用稳态遗传与数组型染色体的方法得到了偏差率小于 0.45% 的四组 TSP 问题最优解。论文第三部分从“算法改进”、“改进评估”、“原因分析”、“测试方法”、“测试中的发现”五个方面做了详细的阐述。最后，给出了四个数据集上的最佳结果展示，包括偏差百分比、相关的参数集设置、最优收敛曲线以及最优路径。

**关键词：**Galib 库遗传算法；TSP 问题；局部最优解；调参测试与评估

## 1 算法功能与意义介绍

本文所述遗传算法针对 TSP 问题设计与实现。TSP 问题（*Traveling Salesman Problem*）又译为旅行推销员问题、货郎担问题，是数学领域中著名问题之一。问题具体描述如下[1]：假设有一个旅行商人要拜访  $n$  个城市，他必须选择所要走的路径，路径的限制是每个城市只能拜访一次，而且最后要回到原来出发的城市。路径的选择目标是要求得的路径路程为所有路径之中的最小值，这即是遗传算法中的目标函数。简单地说就是“已给一个  $n$  个点的完全图，每条边都有一个长度，求总长度最短的经过每个顶点正好一次的封闭回路”。

TSP 问题是一个组合优化问题。该问题可以被证明具有 NPC 计算复杂性。因此，任何能使该问题的求解得以简化的方法，都将受到高度的评价和关注[1]。TSP 问题广泛应用于如物流配送、智能交通控制、电路板钻孔等方面，所以近年来引发了众多学者的关注，并取得了一系列的研究成果。对于较小规模 TSP 问题[2]，精确算法能快速地取得最优解，但难以应对规模较大、结构较复杂的问题。对于较小规模的 TSP 问题，精确算法能快速地取得最优解，但难以应对规模较大、结构较复杂的问题。于是，有学者提出在不能获得全局最优的情况下，寻找高质量的近似解来求解 TSP 问题的启发式算法，包括遗传算法（GA）[3]、粒子群算法、模拟退火算法、禁忌搜索算法、蚁群算法等以及

此类算法的混合算法或改进算法。在遗传算法中，通过产生高质量的初始解可以提高求解问题的速度，甚至改进最终解的质量。此外，遗传算法具有较好的全局搜索能力。

## 2 遗传算法描述

遗传算法 GA(genetic algorithm) 是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法，该算法具有自组织、自适应、自学习和群体进化功能，有很强的解决问题的能力，在许多领域都得到了应用。

### 2.1 基本运算过程

遗传算法基本运算过程的大体流程图如图 1 所示。图中省略了读取、存储、处理初始文件数据和最后输出路径数据到文件两个流程。各个阶段的具体内容如下（具体实现于 tspmain.cpp 文件中）：

1. 初始化阶段：
  - 1.1 定义一组某种类型的染色体（比如 GA1DArrayGenome），设置染色体的编码方式，包括指定计算目标值函数（TSPObjective，即路径长度）、指定初始算子（TSPInitializer）、交叉算子（TSPCrossover）和变异算子（TSPMutator）；
  - 1.2 将染色体应用于某种遗传算法（如 GASimpleGA 或者 GASTeadyStateGA），设置其染色体编码方式，每一代要替换的个体数（nReplacement），总的运行代数（nGenerations），指定将目标函数值最小化的目标（minimize 函数），设置种群大小（populationSize）、变异概率（pMutation）、交叉概率（pCrossover），最后，使用从系统时钟得到的随机种子初始化遗传算法。
2. 个体评价：计算这一代群体中各条染色体的适应度，这一步以及接下来的三步都是在遗传算法的进化函数（step）中实现或者使用，具体不再展示。
3. 选择运算：具体封装在遗传算法类如 GASTeadyStateGA 中，对每代群体创建一个临时的个体群体并将这些添加到以前的群体，然后删除固定个数的最坏的个体（即通过 nReplacement 函数指定的每代替换的个体数），以使群体恢复到原来的大小。
4. 交叉运算：模拟基因重组时染色体交叉片段的方式，交换两个染色体的相同长度的相同片段，但为了避免重复，不能直接交换，在确认交换片段后去除染色体中的重复基因，为交换片段留出空位。
5. 变异运算：模拟基因突变的方式，对单条染色体上的基因进行两种方式的修改（随机选择两个基因交换其位置或者随机选一个基因片段进行反转），通过随机产生一个 0-1 之间的概率数字选择相应的变换。

6. 终止条件判断阶段：达到指定代数时终止遗传，输出最优解，写入文件，在遗传算法类 GASTeadyStateGA 中的 done 函数中实现。

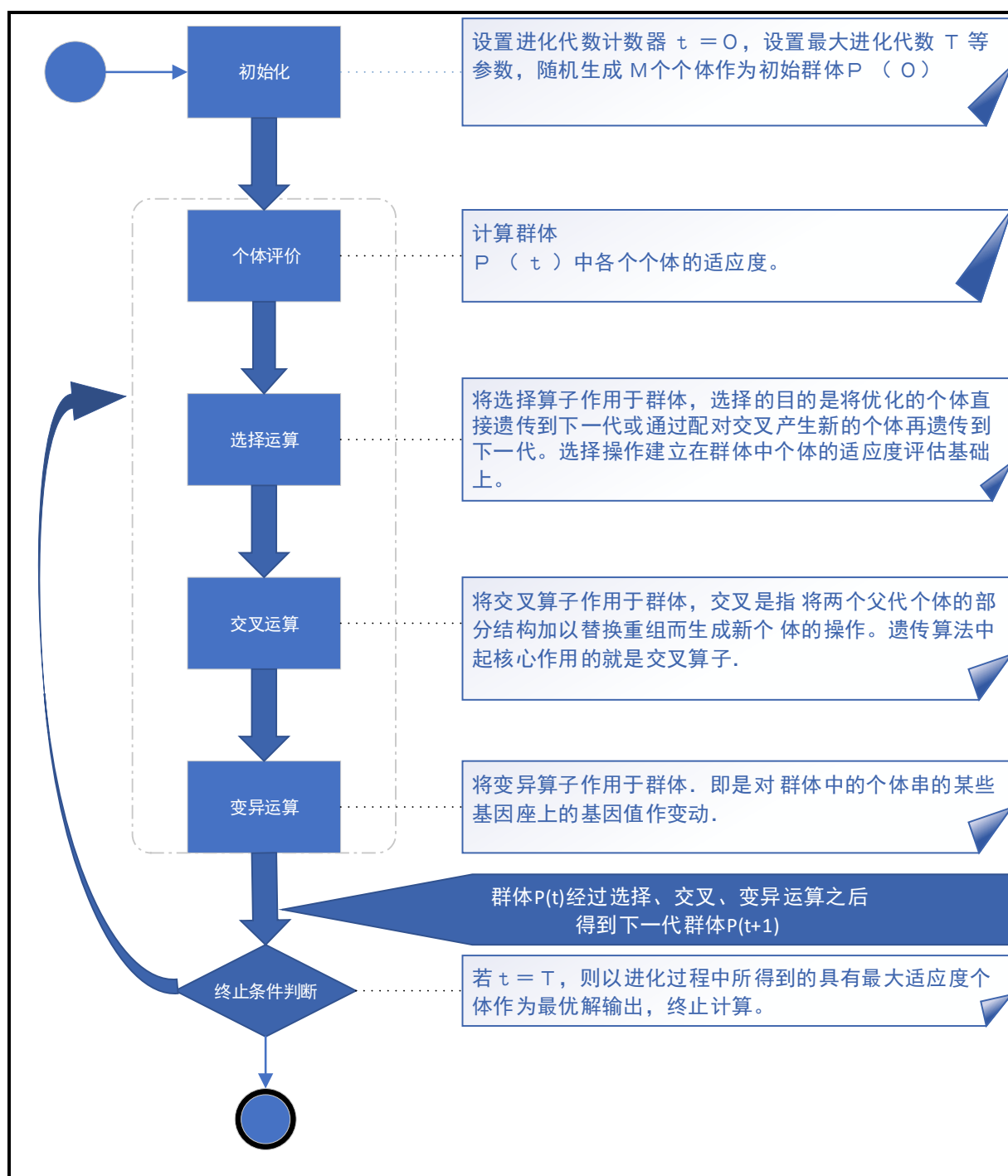


图 1 遗传算法流程图

## 2.2 遗传算法设计

本文基于 Galib2.4 版[4]设计与实现了一个解决 TSP 问题的遗传算法。一个完整的遗传算法包括染色体编码、初始化种群[5]，遗传操作以及适应度计算，针对遗传算法

的主要因素，GAlib 设计实现了遗传算法类、染色体类、种群类、选择方案类及各自的派生体系，下面从遗传算法类和染色体类的角度介绍 GAlib 中的具体设计。

### ● 遗传算法类 (GAGeneticAlgorithm)

GAGeneticAlgorithm 是一个抽象基类，定义于 GABaseGA.h 文件中，声明了遗传算法通用的一些参数、算子、选择子、最大或最小化目标函数等，还包含了变异概率、交叉概率、进化代数等数据，算法内置的进化停止准则包括最大进化代数准则和个体收敛准则[5]。根据群体更新机制的不同，有 4 种遗传算法类继承这个基类，分别是：

1. 标准遗传算法 (GASimpleGA)：仅保留父代最佳个体或不保留父代群体中任何个体，子代群体完全由交叉变异等遗传操作产生；
2. 稳态遗传算法 (GASteadyStateGA)：子代群体由进化操作产生的部分新个体和父代群体中部分个体按一定比例共同组成；
3. 增量遗传算法 (GAIcrementalGA)：每代进化操作只产生一个或两个新个体，新个体替换其父代群体中的某些个体；
4. 种群遗传算法 (GADemeGA)：该算法同时控制数个群体进行演化，每个群体的进化采用稳态遗传算法，群体之间进行交换个体操作。

相关类的类图继承关系如下：

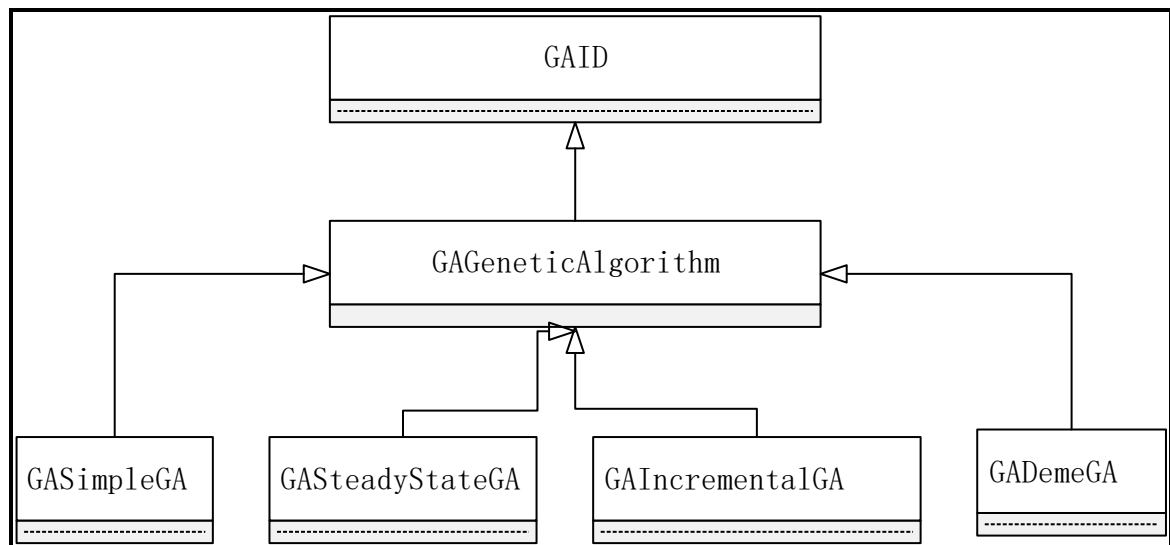


图 2 遗传算法类继承类图

### ● 染色体类 (GAGenome)

染色体是实际问题中相关参数的载体，本质上是一种表示问题的解的统一的数据结构。GAGenome 是一个抽象基类，定义了一些常量（如维度 Demension 中的 LENGTH、WIDTH、HEIGHT、DEPTH 分别为 0, 0, 1, 2）和函数原型（包括 Evaluator、Initializer、Mutator、crossover 等）及其派生类的函数原型。染色体是遗传算法主要作用的对象，可以在染色体上进行初始化、变异、交叉、评估和比较操作。GAlib 中主要实现了四种染色体，每种染色体继承自 GAGenome 和一种对应的数据结构类，简述如下：

1. 链表型染色体 (GAListGenome)：继承自 GAGenome 和 GAList，可以用于基于顺序的表示或可变长度序列，以及传统的列表应用。使用时必须为此类定义初始化运算符。因为默认的初始值设定项是“NoInitializer”。主要的遗传操作有：SwapMutator, DestructiveMutator, OnePointCrossover, OrderCrossover, PartialMatchCrossover, CycleCrossover。
2. 树型染色体 (GATreeGenome)：继承自 GAGenome 和 GATree，可用于直接操纵树对象，树对象包括二叉树和非二叉树，使用时也要为此类定义初始化运算符。主要的遗传操作符有：DestructiveMutator, SwapSubtreeMutator, SwapNodeMutator, OnePointCrossover。
3. 数组型染色体 (GAArrayGenome)：是最大的一个染色体类族，根据维度的不同，分 GA1DArrayGenome, GA2DArrayGenome, GA3DArrayGenome 三种，继承自 GAGenome 和 GAArray 类，数组元素都是基因。引入等位基因 (allele) 的概念，GA1DArrayAlleleGenome, GA2DArrayAlleleGenome, GA3DArrayAlleleGenome 继承自相应的没有引入等位基因的染色体类。
4. 二进制串型染色体 (GABinaryString)：二进制字符串对象是一个简单的位字符串的实现，可以调整字符串大小。每一个比特都由一个内存的字 (a single word of memory) 来表示。主要的成员函数有：copy, bit, equal, move, set/unset, size, resize, randomize。依据染色体维度的不同，GA1DBinaryStringGenome、GA2DBinaryStringGenome、GA3DBinaryStringGenome 继承自 GABinaryString 和 GAGenome 类。

相关类的类图继承关系如图 3 所示。

### 3 算法改进与效果测评

#### ■ 算法改进

算法改进主要体现在对不同遗传算法、染色体类的使用、调用方面，尝试了四种遗传算法和四类染色体类的方法。主要代码更改部分如下所示：

➤ 四种遗传算法的尝试：

```
// 每一代只产生2个，替换上一代最差的
GASteadyStateGA ga(genome); ga.nReplacement(2); ga.nGenerations(500000);
//GAIncrementalGA ga(genome); ga.nGenerations(500000);
//GADemeGA ga(genome); ga.nReplacement(4); ga.nGenerations(500000);
//GASimpleGA ga(genome); ga.elitist(gaTrue)
```

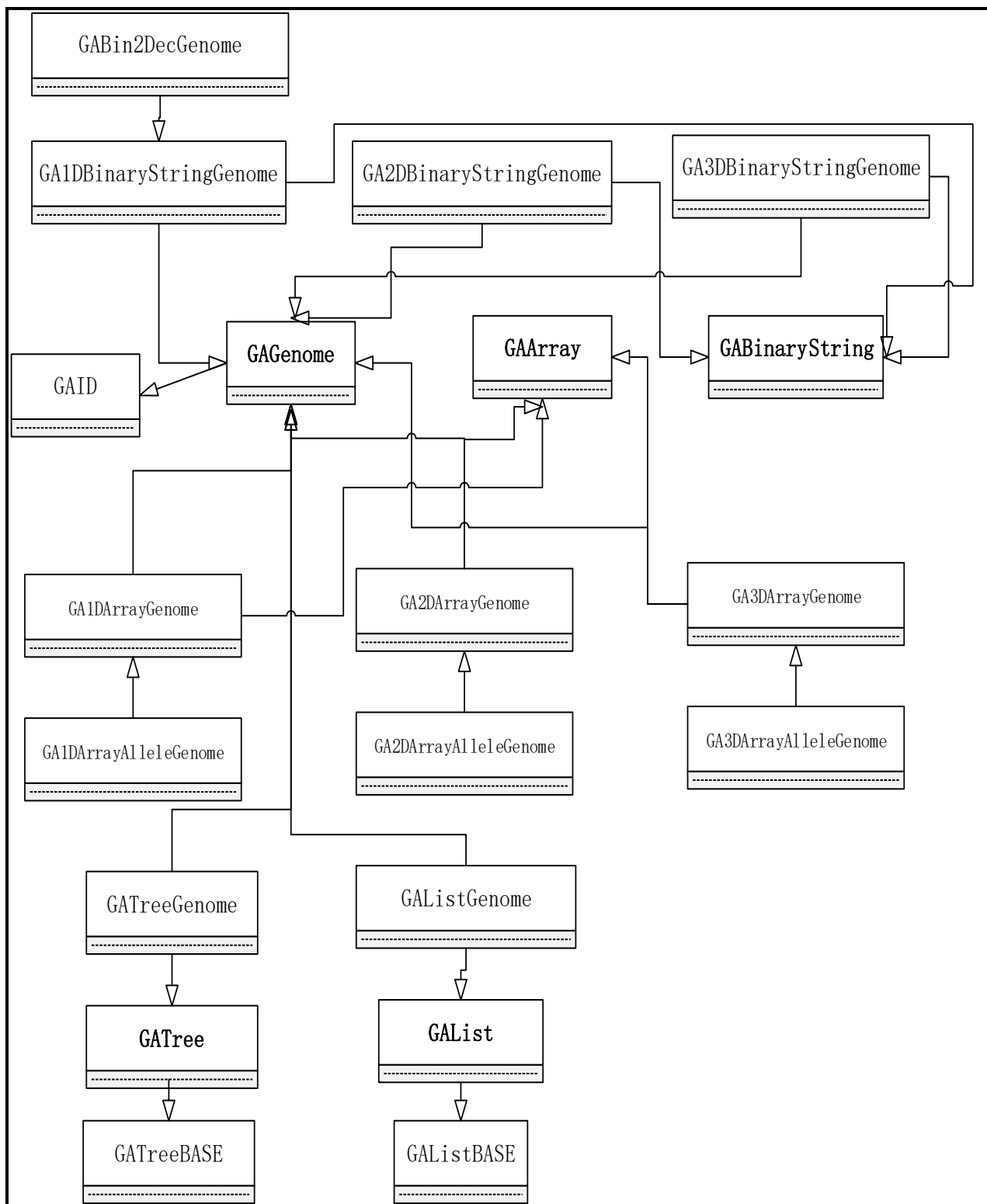


图 3 染色体类继承类图

➤ 四类染色体类方法的尝试:

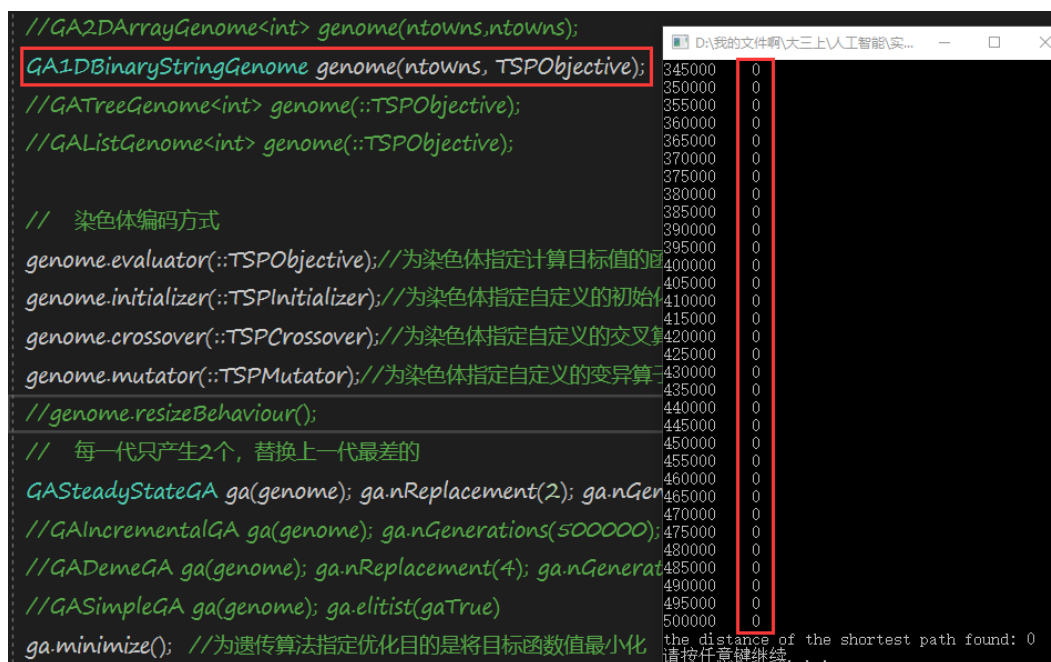
```
//定义TSP问题的编码方案为一维的整数数组，其固定长度为城市个数
GA1DArrayGenome<int> genome(ntaxons);
//GA1DArrayAlleleGenome<int> genome(ntaxons, GAAlleleSet<int>(1,100,5));
//GA2DArrayGenome<int> genome(ntaxons,ntaxons);
//GA1DBinaryStringGenome genome(ntaxons, TSPObjecive);
//GATreeGenome<int> genome(::TSPObjecive);
//GAListGenome<int> genome(::TSPObjecive);
```

其他部分基本不变，每次改进后分别在四个数据集(berlin52, lin105, rat99, pr76)上进行调整参数步骤（即效果测评），观察这种方法、数据结构的效果。

## ■ 改进评估

由于经验不足、时间有限，笔者没有整理、记录好每种方法的效果，只在某个数据集上得到突破性进展时记录相关参数、输出文件。虽然做了上千次的测试，但同时也损失了很多数据，因此不能以表格形式明了的展示出相关数据，只能概括大致的效果。

总体来说，几乎所有的最优解都是在 GASTeadyStateGA 类和 GA1DArrayGenome 类的基础上调参实现的，其他方法上：有一部分全部输出 0（GA1DBinaryStringGenome、GATreeGenome、GAListGenome），如图 4；有一部分和未改之前基本一样，看不出明显的变化或者性能有所下降（如引入了等位基因 Allele 的染色体 GA1DArrayAlleleGenome 时、使用 GAINcrementalGA、GASimpleGA 算法时），如图 5；还有些方法运行时产生 bug（比如产生内存访问错误、中断、某个标识符未定义等等），如图 6、图 7。



The screenshot shows a code editor with C++ code for a Genetic Algorithm. A red box highlights the line `GA1DBinaryStringGenome genome(ntaxons, TSPObjecive);`. To the right, a console window displays a list of 50,000 zeros, indicating that the distance of the shortest path found is 0 for all iterations. The code includes comments in Chinese explaining the purpose of various methods like `genome.evaluator`, `genome.initializer`, `genome.crossover`, and `genome.mutator`. The console output at the bottom reads: "the distance of the shortest path found: 0 请按任意键继续..."

图 4 采用 GA1DBinaryStringGenome 时的全 0 结果



```

GA1DArrayGenome<int> genome(ntaxons);
//GA1DArrayAlleleGenome<int> genome(ntaxons, GAAlleleSet);
//GA2DArrayGenome<int> genome(ntaxons,ntaxons);
//GA1DBinaryStringGenome genome(ntaxons, TSPObjective);
//GATreeGenome<int> genome(::TSPObjective);
//GAListGenome<int> genome(::TSPObjective);

// 染色体编码方式
genome.evaluator(::TSPObjective); //为染色体指定计算目标值的函数
genome.initializer(::TSPInitializer); //为染色体指定自定义的初始化器
genome.crossover(::TSPCrossover); //为染色体指定自定义的交叉算子
genome.mutator(::TSPMutator); //为染色体指定自定义的变异算子
//genome.resizeBehaviour();
// 每一代只产生2个, 替换上一代最差的
//GASteadyStateGA ga(genome); ga.nReplacements(2); ga.nGenerations(500000);
GAIIncrementalGA ga(genome); ga.nGenerations(500000);
//GADemeGA ga(genome); ga.nReplacements(4); ga.nGenerations(500000);

```

the distance of the shortest path found: 1250  
请按任意键继续...

图 5 使用 GAIIncrementalGA 时无明显变化（多次测试没有突破进展，rat99 数据集）

```

154 GA2DArrayGenome<T>::resizeBehaviour(GAGenome::Dimension which) const {
155     int val = 0;
156     if(which == WIDTH) {
157         if(maxX == minX) val = FIXED_SIZE;
158         else val = maxX;
159     }
160     else if(which == HEIGHT) {

```

错误列表

代码	说明	项目	文件	行
C4996	'strcpy': This function or variable may be unsafe. Consider using strcpy_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.	tsp_ga	gastatistics.h	164
C2065	"FIXED_SIZE": 未声明的标识符	tsp_ga	ga2darraygenome.cpp	157
C2065	"FIXED_SIZE": 未声明的标识符	tsp_ga	ga2darraygenome.cpp	161

图 6 使用 GA2DarrayGenome 类时的标识符未声明错误

```

64 if(n == sz) return sz;
65 T* tmp = (n ? new T[n] : 0);
66 for(int i=((n < sz) ? n : sz); i-->0; i--)
67     delete [] a;
68 a = tmp;
69 return sz=n;
70 }
71 int equal(const GAArray<int> &a, const GAArray<int> &b)
72 {
73     unsigned int dest, u;
74     for(unsigned int i=0; i<a.GetSize(); i++)
75         if(a[i] != b[i]) return 0;

```

已引发异常  
引发了异常: 读取访问权限冲突。  
this 是 nullptr。

复制详细信息

异常设置

- ☒ 引发此异常类型时中断
- 从以下位置引发时除外:
  - ☐ tsp\_ga.exe

打开异常设置 | 编辑条件

局部变量

名称	值	类型
this	0x00000000 <NULL>	GAArray<int> *
__vfptr	<无法读取内存>	void **
sz	<无法读取内存>	
a	<无法读取内存>	
i	变量已被优化掉, 因而不可用。	unsigned int
n	99	
tmp	变量已被优化掉, 因而不可用。	

图 7 使用 GADemeGA 算法时的无法读取内存错误

## ■ 原因分析

某些方法效果不佳的主要原因是笔者没有详细了解整个遗传算法的构造，不清楚其中的具体细节，导致不能正确、合理、有效地使用 GaLib 库提供的方法，得不到正常的或者更优的路径。其次是因为在原有方法（即使用 GASteadyStateGA 和 GA1DarrayGenome）上经过很多次调参测试，已经在某种程度上达到了遗传算法的上限值。因为遗传算法是面向“结合性能上的考虑而允许缺失一小部分最优路径”的算法，本身就不是以求最优解为必要目标。所以当原有方法经过大量测试达到某种上限后用其他方法也很难有突破。

## ■ 测试方法

测试方法主要就是调节各个参数的大小，在某种环境、某套代码下求得局部最优值，使用时一定要注意变量唯一、控制变量、明确调整目标。

针对本文所述 TSP 问题，重点调节参数有 3 个：种群大小、变异概率、交叉概率；次要调节参数有 2 个：每代要替换的个数（有的方法不需要指定这个参数，如 GAIIncrementalGA）、总遗传代数。调整次要参数起辅助作用，最好保持总替换/搜索个体数不变，即一百万。尝试过  $2*500000$ 、 $4*250000$ 、 $10*100000$ 、 $20*50000$  等，总的说影响并不大，所以我是先固定了这两个辅助参数为  $2*500000$ ，然后重点调节其他三个参数（不过后来发现调节这两个辅助参数也能得到不错的结果，但由于时间有限，仍只选择了 3 个重点参数进行测试）。

调参时大致步骤如下：

- 1) 固值：选择并固定种群大小（以求得在这一种群大小下的局部最优值）；
- 2) 初调：根据经验选择变异概率值、交叉概率值进行初步调试，确定大致范围；
- 3) 细化：逐步减小变异概率、交叉概率的“改变步长”，即每次改变参数时增加或减小的大小，一般先固定一个概率，确定另一个概率在某一可接受精度下的最优值；
- 4) 迭代：记录本次迭代下种群数量、作用的数据集、两种概率分布等相关信息然后改变种群大小，继续进行前三步，直到产生一个在 0.5% 的偏差下的最优解而且较长时间内不再有突破（减小随机的影响以使最优解稳定）。

## ■ 测试过程中的一些发现

经过对三个数据集（lin105、pr76、rat99）的大量测试，逐渐感受或者发现了一些不是很科学的规律、特征或者测试方法，列出如下：

1. 多个局部最优解的存在：一种方法作用于一个数据集上，可能因参数的不同有不同的最优解，即局部最优解，只需固定某些参数（如种群数量、两个

辅助参数) 然后调整剩下的参数, 求得在局部条件下的最优值。不同的局部有不同的最优解, 这就需要大量的调参测试和测试数据的记录。

2. 概率范围分布: 经过大量测试, 发现交叉概率设置为 0.21 左右、变异概率在 0.01、种群在 100-120 时取到最佳的局部最优解。当然这也只是目前一个最佳的局部最优解, 其他地方(比如种群在 260、交叉概率为 1、变异概率为 0.02 时)也有局部最优解, 也可能还有未知的地方有更佳的局部最优解, 甚至能达到数据集上的全局最优解。此外, 变异概率的变化范围远小于交叉概率, 在 0.01-0.02 左右波动, 而交叉概率跨 0.2-1.0 都能有比较好的局部最优解。
3. 概率影响规律: 经测试发现, 变异概率的变化范围小, 但影响大, 只需变动千分位的量级就可能对结果有较大的影响, 而有时候在某个范围内变动对结果完全无影响, 只要突破那个范围, 结果立刻有大的改变。对于交叉概率, 变化范围大, 对结果的影响也比较平稳。此外, 种群大小、交叉概率、变异概率三者似乎存在某种制约关系, 就像三条边, 只有满足某种关系时才能组成一个三角形, 而局部最优解就是当三者构成的三角形是具有某些特征的三角形(比如等腰、直角、等边等等)。粗略地估计, 当种群比较大时交叉概率也需比较大, 而变异概率也可能相应的增大一点点。
4. 重现困难(随机性): 最好的局部最优解一般难以再现, 或是今天每次运行都是最好的局部最优解, 但明天同样的代码就无论如何跑不出来了。
5. 多次运行结果单一化(非随机性): 在 Visual Studio 中按“Ctrl+F5”组合键运行时往往得到相同的结果, 但用调试器运行时往往有随机性的不同结果, 一般前者比后者的路径长度更短, 但也存在后者更短的时候。此外, 每当参数值有变动、第一次运行的时候, 是更短的路径出现的时候, 当多次按“Ctrl+F5”后, 运行结果的最终路径会迅速(1-3 次内)稳定下来, 稳定后每次的所有输出结果都完全一样。
6. 收敛快: 总体来说这套遗传的收敛是很快的, 不需要改变总的搜索个体数, 将之固定在一百万是足够了, 继续遗传更多代数(五百万、八百万等)结果基本都保持不变了。
7. 大胆尝试: 为了得到突破路径, 需要大胆的调参尝试, 一次性根据经验改变三个主要参数的值, 用这种方法可能得到意想不到的结果, 因此, 不要在某个局部死死地求最优解, 当似乎走入了死胡同时应立刻改变求解方向, 寻求更加的局部。
8. 取值小技巧: 种群大小最好是以 0 结尾, 即 10 的倍数, 而且必须是偶数。

#### 4 最佳结果展示

本文用遗传算法对“berlin52”、“pr76”、“rat99”、“lin105”四个数据集进行了TSP问题的测试，得到的最优解皆在0.5%的偏差以下，如表1所示。

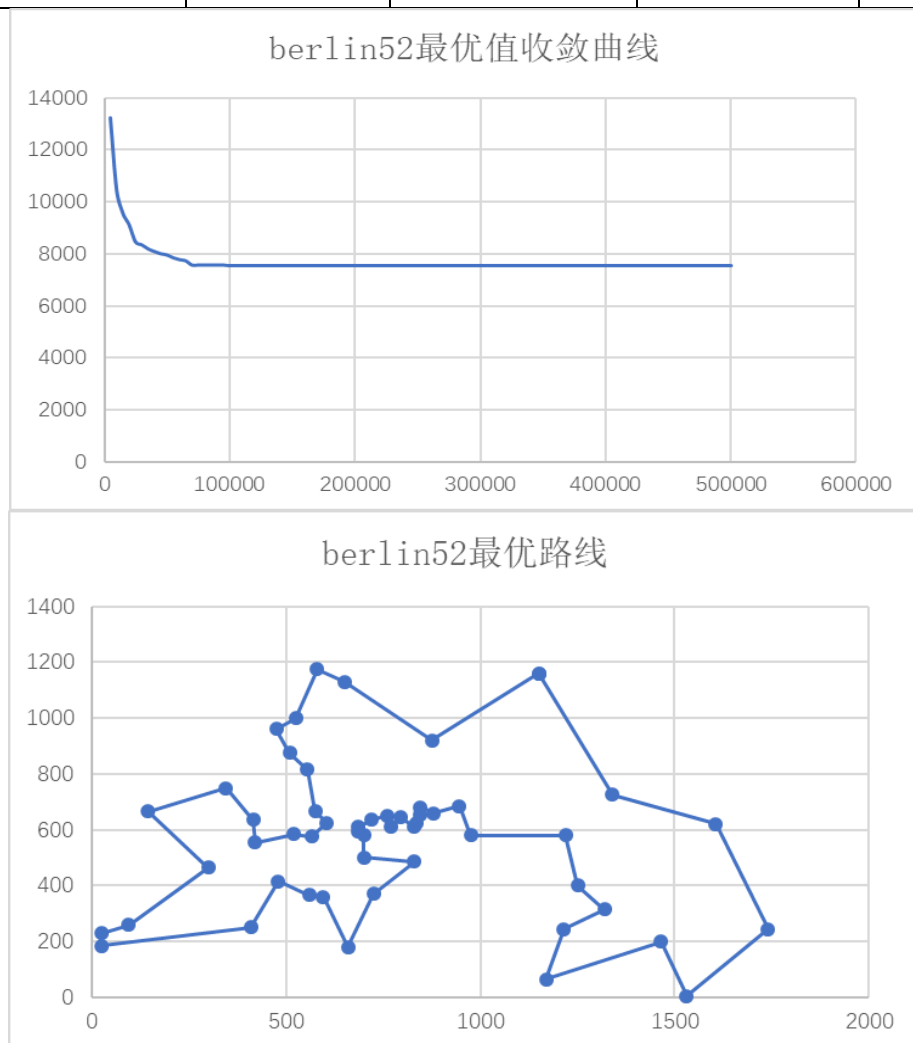
数据集	理论最优路径长度	实际求得最优路径长度	偏差百分比
berlin52	7542	7542	0
pr76	108159	108280	0.112%
rat99	1211	1216	0.413%
lin105	14379	14447	0.437%

表1 四个数据集的最优解与偏差

四个数据集的详细结果如下所示。

##### ◆ berlin52:

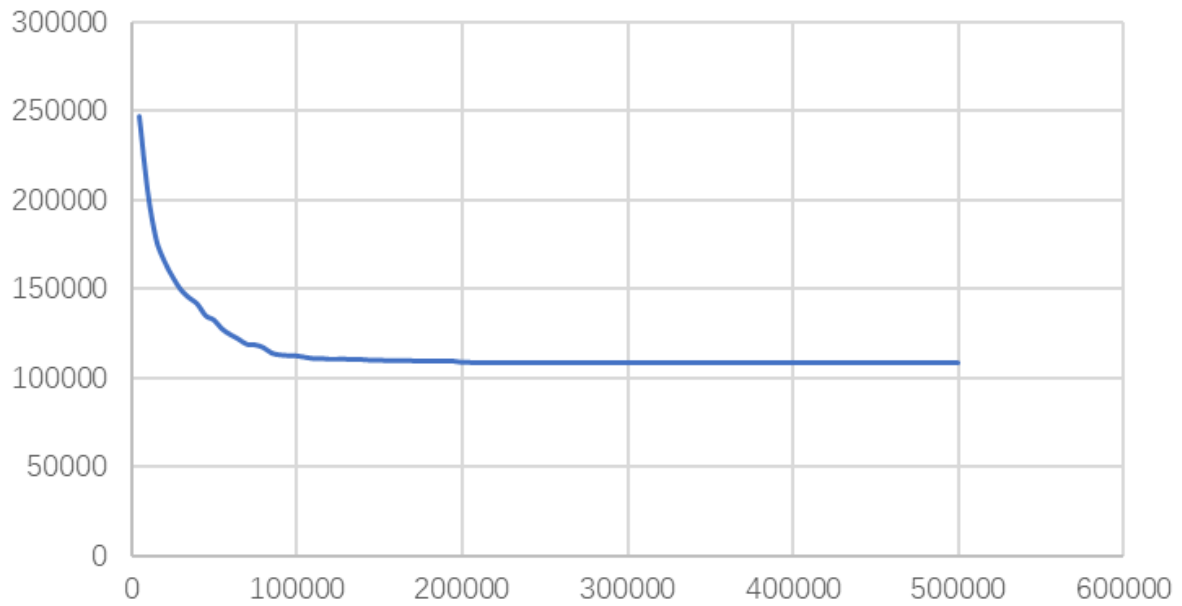
种群大小	进化代数	每代替换个数	交叉概率	变异概率
200	500000	2	1.0	0.02



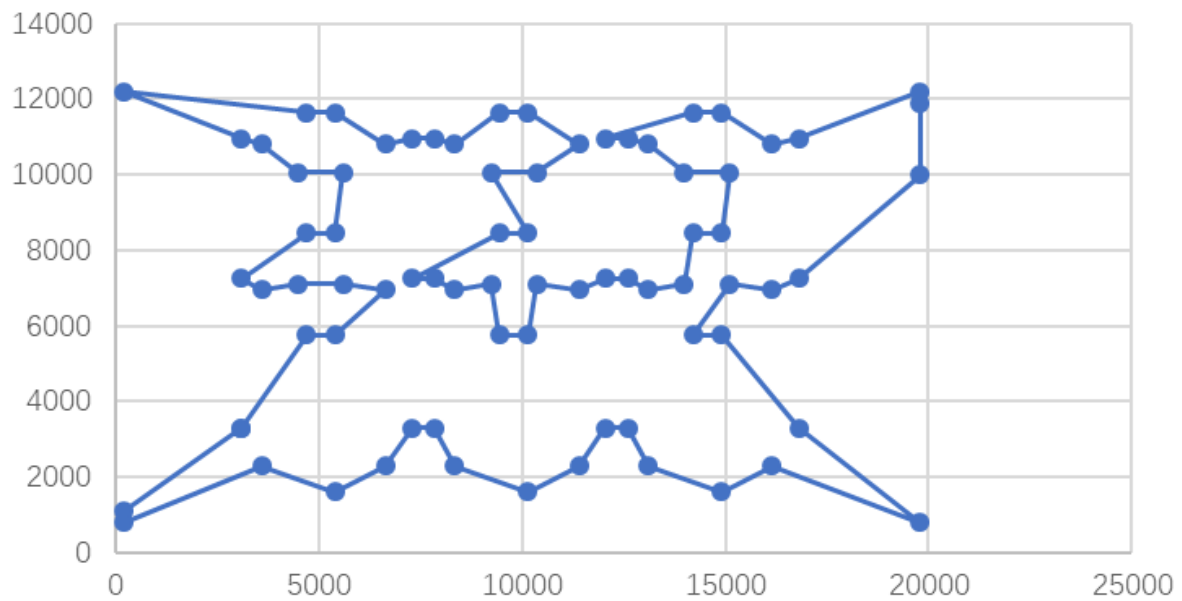
◆ pr76:

种群大小	进化代数	每代替换个数	交叉概率	变异概率
122	500000	2	0.01	0.21

pr76最优值收敛曲线

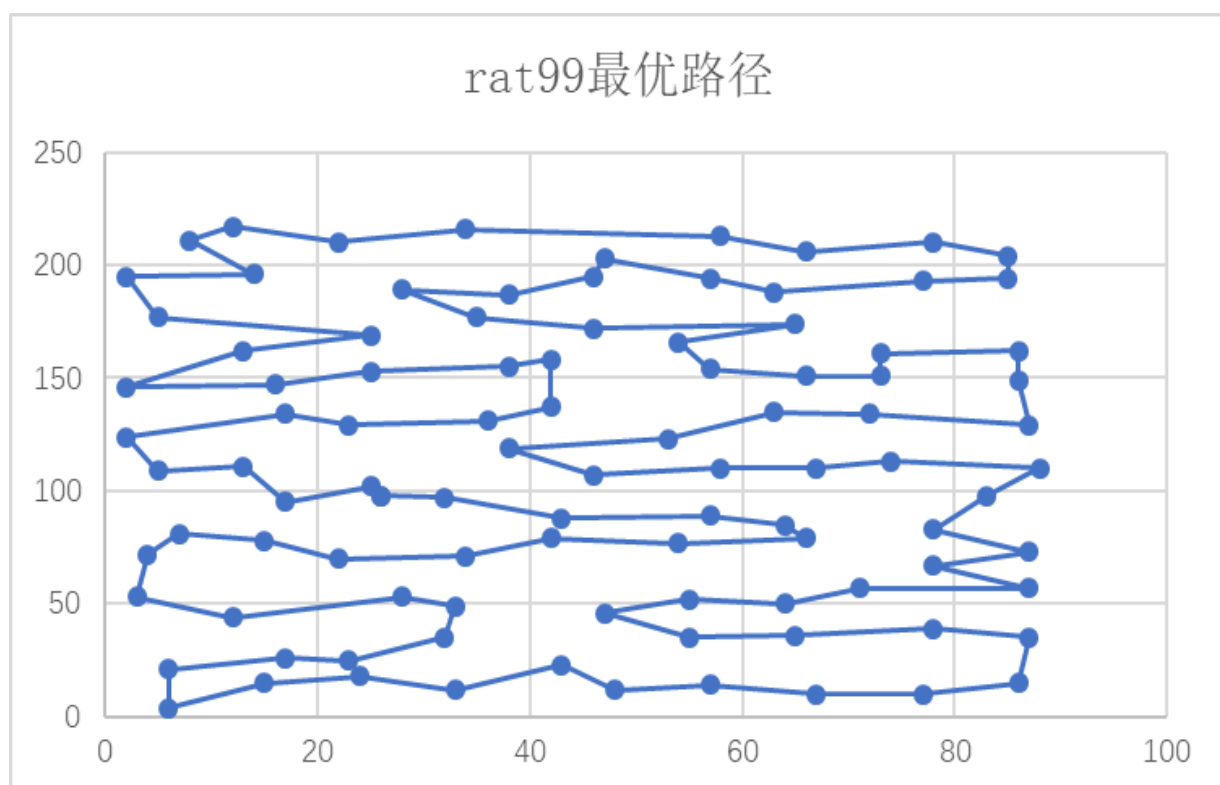
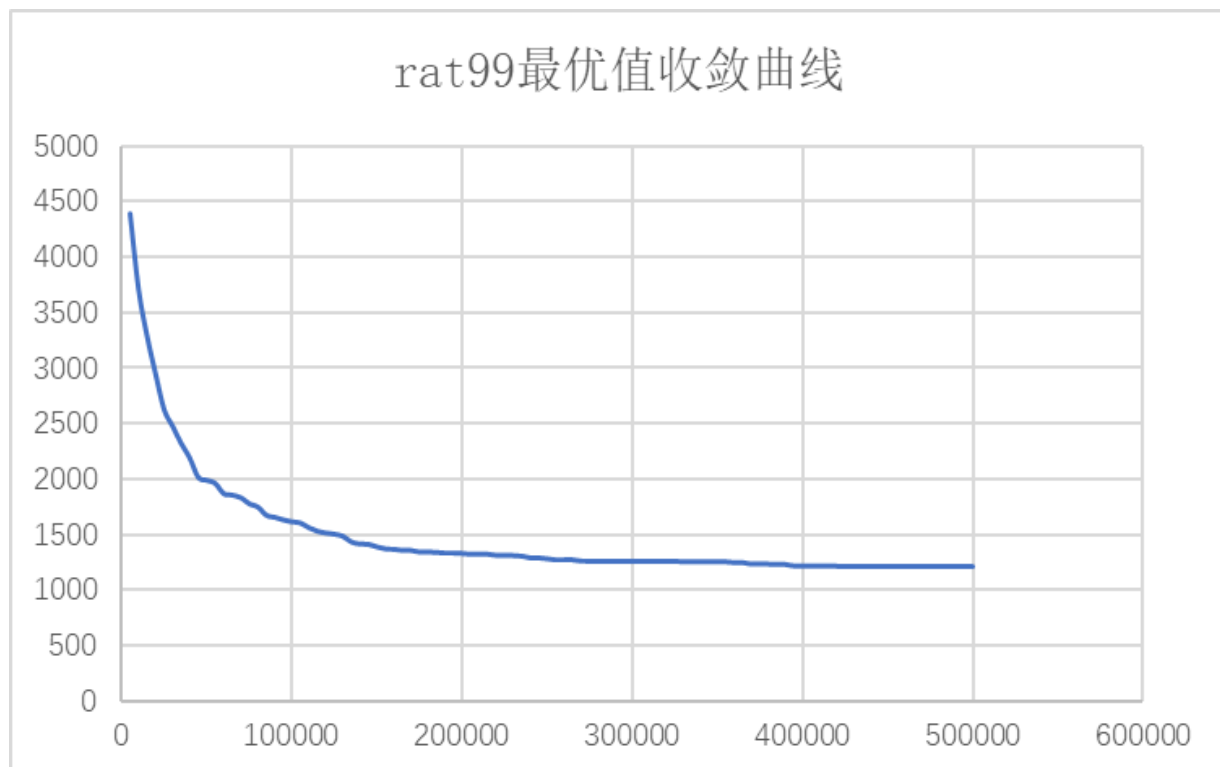


pr76最优路径



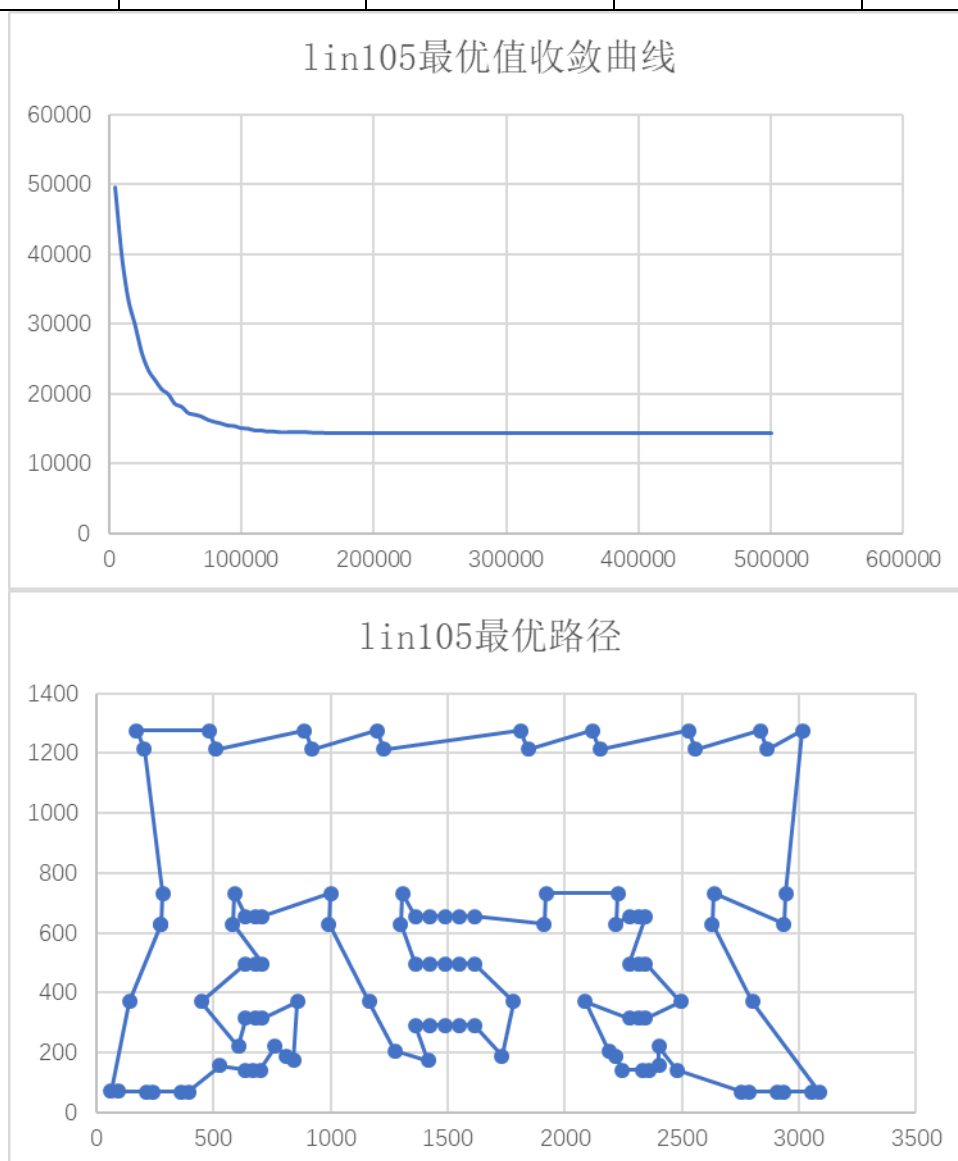
◆ rat99:

种群大小	进化代数	每代替换个数	交叉概率	变异概率	路径长度
100	500000	2	0.01	0.21	1216
260	500000	2	0.02	1.0	1236



◆ lin105:

种群大小	进化代数	每代替换个数	交叉概率	变异概率
100	500000	2	0.01	0.21



## 5 结语

“任何非常先进的技术，初看都与魔法无异。”[6]看似魔法的遗传算法程序设计也都不过是由六步构成：求通过遗传算子进行转换的结构集；求适问题域的初试结构集；评估结构的适应度测定；求转换结构的遗传算子集；描述每一代成员的参数和状态；确定终止条件集合。希望将来有更多优秀、神奇的遗传算法问世，为人类的生活解决更多、更广泛的复杂问题，造福人类。

## 6 参考文献

- [1] 孙和军. TSP 问题[EB/OL]. (2018-06-07) [2018-12-25]. [https://baike.baidu.com/item/TSP 问题/840008?fr=aladdin](https://baike.baidu.com/item/TSP问题/840008?fr=aladdin).
- [2] 张玉州,梅俊,徐廷政.一种求解 TSP 问题的混合遗传算法[J].安庆师范大学学报,2018,24(3):77-81.
- [3] 刘荷花,崔超,陈晶.一种改进的遗传算法求解旅行商问题[J].北京理工大学学报,2013,33(4):390-393.
- [4] Matthew Wall. GALib: A C++ Library of Genetic Algorithm Components[M] Massachusetts Institute of Technology,1996.
- [5] 张泰忠,徐成.Galib 在遗传算法实现中的应用[J].计算机系统应用,2011,20(10):184-188
- [6] George F.Luger.Artificial Intelligence: Structures and Strategies for Complex Problem Solving[M].郭茂祖,刘扬,玄萍等,译.北京:机械工业出版社,2009.