/基于近似算法的TSP问题求解

姓名学号： 王程钥 17341146
联系方式： wangchy56@mail2.sysu.edu.cn
日期： 2020-7-12

摘要 本文分别使用局部搜索，模拟退火算法，遗传算法去解决TSP问题。对于局部搜素哦算法，本文尝试了多种邻域计算策略。对于遗传算法，本文使用了多种训练策略，并引入了额外信息。本文选择TSPLIB[1]中的berlin52问题进行实验，在最优策略下汉密尔顿回路的最短距离可以达到7544，和最优解7542非常接近。

1 导言

TSP问题（Traveling Salesman Problem）是一个非常经典的组合优化问题。这里首先简单地复述一下这个问题。有一个旅行的商人要拜访n座城市，他从某一座城市出发拜访所有城市，每座城市必须被拜访一次且仅能被拜访一次，最后要回到原来出发的城市，问如何规划路径才能使整条路径的距离之和尽量小。形式化地说，就是给你一张有n个点的无向连通完全图，你要找到一条距离最短的汉密尔顿回路。

TSP问题很早以前就被证明是NPC问题（NP-Complete Problem），即它并没有多项式级别时间复杂度的解法。如果要保证正确性，这里可以使用dfs搜索的方法或状态压缩动态规划的方法，最优时间复杂度大致是$O(2^n * n)$。为了处理更大规模的TSP问题或其它NPC问题，我们引入了一些近似算法，比如局部搜索法，模拟退火法，遗传算法等。这些算法并不能保证找到问题的最优解，但它通常能够至少找到一组接近最优解的近似解。

局部搜索算法（Local Search）是一种基于贪心的组合优化算法，它可以快速地找到一个局部最优解，在凸问题其具有正确性。模拟退火算法（Simulated Annealing）则是局部搜索算法的一个改进，它引入了模拟退火的机制，可以帮助局部索索算法跳出局部最优解。在本文中，我首先尝试了基于局部搜索的近似算法，采用了基于两点和基于三点的邻域策略进行训练。随后我尝试在局部搜索的基础上引入模拟退火的思想，从而一定程度上解决局部最优解的问题。

遗传算法（Genetic Algorithm）则是一种仿生学算法，它模拟生物遗传过程中交配，变异等过程，利用遗传学中优胜劣汰适者生存的原理进行最优化。在本文中我设计了基于片段的交叉操作，并使用局部搜索中寻找邻域的方法替换遗传算法中的变异操作。之后，我又尝试引入热身训练（warm-up trainning）的机制，也尝试在变异的时候引入一些额外信息。最终经过优化的遗传算法取得了不错的效果。

2 实验过程

2.1 局部搜索

2.1.1 算法简介

局部搜索算法又称爬山法，是一种基于贪心的组合优化算法，和深度学习中的梯度下降也有雷系的原理。局部搜索算法首先随机生成一个起始点作为搜索的出发点，每次在当前点附近的邻域内使用估价函数选择一个最优的邻域点（在梯度下降法中，就是梯度最大的方向），并前往那个邻域点。局部搜索算法会一直重复这个操作，直到进入稳定状态或搜索达到一定次数。

对于TSP问题，初始解就是n座城市的一个全排列，邻域就是对排列进行一次修改操作，股价函数(1)就是整条路经的路程和。算法的具体流程见下面的伪代码。

$$E(Route) = \sum_{i=1}^{n} dis(Route_i, Route_{(i+1) mod n}) \tag{1}$$

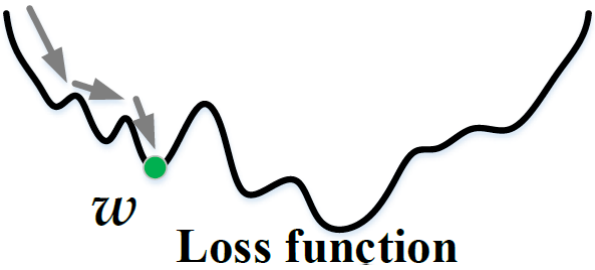
```
算法1：局部搜索算法
输入：估价函数E，搜索步数Steps
输出：最优路径Route
Route := Random Initialize Permutation
for i in [0, Steps] do
    Route_best := Route
    for j in [0, 100] do
        Route' := Route邻域中的一条路径
        if E(Route') < E(Route_best) do
            Route_best = Route'
    if E(Route_best) < E(Route)
        Route = Route_best
```

2.1.2 邻域的选择

1. 基于两点的策略
   在一条路径中随机选择两个点x,y（不妨设x<y），要求访问x点后下一个访问的点是y。这里可以采用两种策略，一是在访问x点后先访问y点后的城市，后再访问$x-y$的一段。另一种策略是先访问x点后先倒序访问x~y的一段，后再访问y点后的城市。
2. 基于三点的策略
   在一条路径中随机选择三个点x,y,z（不妨设x<y<z），要求访问x点后下一个访问的点是y，且$x-y$一段再访问z点后访问。这里采用的具体策略如下，再访问x点后访问$y-z$的一段，后访问$x-y$一段，最后访问z之后的一段。对于x~y的一段，根据顺序访问和倒序访问同样分为两种策略。

2.2 模拟退火

使用局部搜索算法可以快速地找到一个比较优秀的点，但其基于贪心算法的本质也会导致其产生一个问题。因为TSP问题显然不会是一个凸问题，所以使用局部搜索很容易导致陷入局部最优解的情况的发生。因此我们需要引入一种机制，让算法有能力再一定情况下跳出局部最优解，找到全局最优解。



模拟退火算法模拟物理退火的机制，首先设定初始温度T，温度T会随着算法的进行逐步下降。在寻找邻域的过程中，如果邻域优于当前解则更新，如果劣于当前解，也可以根据温度和与当前解的差值以一定概率更新。假设当前答案为R，邻域答案为R'更新公式可以参考(2)。在本次实验中，我使用公式(3)作为温度T的变化公式。

$$UpdateProb(R,R') = \begin{cases} 1 & E(R) >= E(R') \\ e^{\frac{E(R)-E(R')}{T}} & E(R) < E(R') \end{cases} \tag{2}$$

$$T_i = \frac{T_0}{log(1+i)} \tag{3}$$

2.3 遗传算法

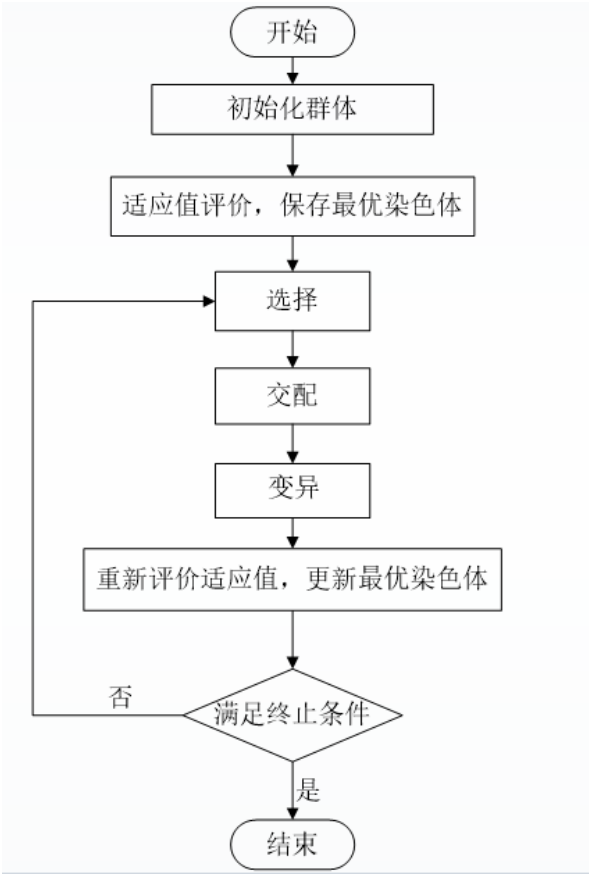
2.3.1 算法简介

遗传算法是一种群体算法，它模拟生物群体繁衍过程中交叉、变异的过程，基于生物学优胜劣汰适者生存的思想进行组合优化。遗传算法可以用来解决TSP问题，下面是一些TSP问题和遗传算法的转化。

- 染色体：一条汉密尔顿回路。
- 群体：多条汉密尔顿回路。
- 染色体的编码方式：一条路径上经过的所有城市的编号构成的排列。
- 群体初始化：随机生成若干全排列。
- 估价函数：基于汉密尔顿回路长度的倒数，参考公式(4)。
- 选择算子：将估价函数归一化生成一个概率分布，依照概率分布进行选择。
- 交配算子：根据概率选择进行交配的一些染色体，将这些染色体两两配对。对于每一对交配的染色体，选择一段长度不大于10的区间进行互换。因为要保证交配后生成的染色体还是一个排列，在完成互换后还要根据情况调整互换区间外的数字。遍历互换区间内的数字构建一个映射表，对于区间外的数字，则要根据映射表进行调整。交换算子的具体操作可以参考下图的例子。映射表的构造方法见算法2。
- 变异算子：对于每条染色体以一定概率发生变异。变异的过程与局部搜搜算法中寻找邻域的过程相同。

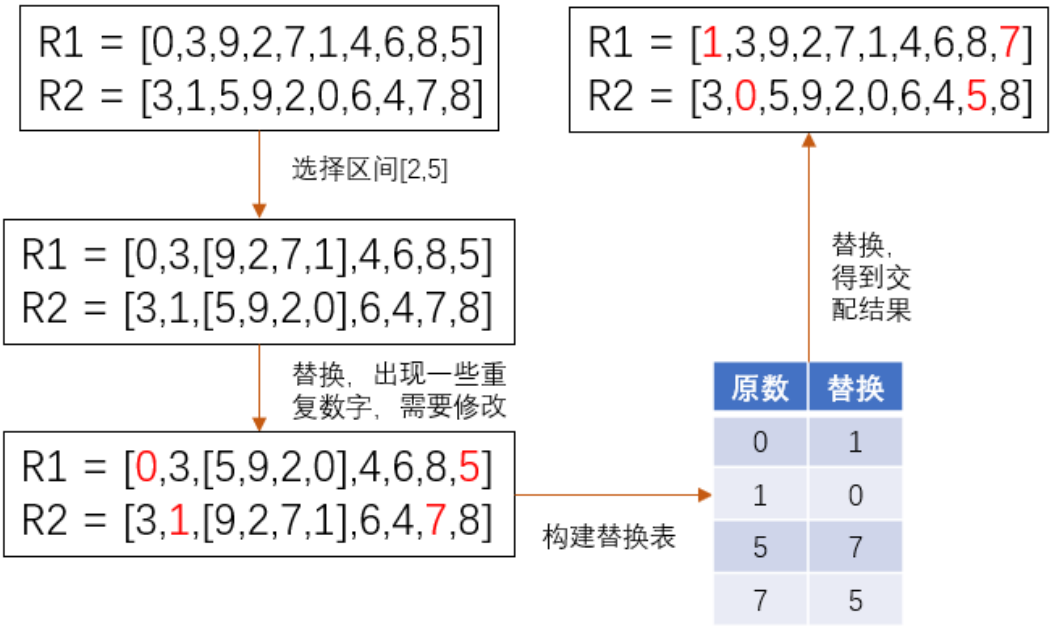
$$Eval(R) = \frac{1}{E(R)^4} \tag{4}$$

以下为遗传算法的流程图和伪代码。




```
/*
   P(t)表示某一代的群体，t为当前进化代数
   Best 表示目前已找到的最优解
*/
Procedure GA
begin
      t←0;
      initialize(P(t));                    //初始化群体
      evaluate(P(t));                      //适应值评价
      keep_best(P(t));                  //保存最优染色体
      while (不满足终止条件) do
      begin
         P(t)← selection(P(t));          //选择算子
         P(t)← crossover(P(t));          //交配算子
         P(t)← mutation(P(t));           //变异算子
         t←t+1;
         P(t)←P(t-1);
         evaluate(P(t));
         if(P(t)的最优适应值大于Best的适应值)
           //以P(t)的最优染色体替代Best
           replace(Best);
         end if
      end
end
```

以下是交换算子的一个例子。



以下是映射表构造的伪代码。

```
算法2：构造映射表
输入：两条路径R1，R2，区间[L，R]
输出：映射表to
num := 所有只在R1[L，R]出现或只在R2[L，R]出现的数字
pos := 所有只在R1[L，R]出现或只在R2[L，R]出现的位置

for po，val in pos do
    while True
        if val in R1 then
            if R2[po] in num then
                to[val] = R2[po]
                break
            else: po = R2[po]在R1中出现的位置
        else
            if R1[po] in num then
                to[val] = R1[po]
                break
            else: po = R1[po]在R2中出现的位置
```

2.3.2 热身训练机制

如果直接使用随机数作为遗传算法的初始值，那么算法的更新是很慢的，也很难能够运行出比较优秀的解。因此我们这里通过动态调节参数的办法进行热身训练（Warm-up Trainning），让随机初始化的种群能够快速达到一个比较优秀的状态，后再使用常规参数运行遗传算法。这里以前200步作为热身训练时间，主要调整两个参数，一是估价函数(5)，二是基因变异的概率(6)。

$$Eval(R) = \begin{cases} \frac{1}{E(R)^8} & step < 200 \\ \frac{1}{E(R)^4} & step >= 200 \end{cases} \tag{5}$$

$$Prob_{Variation} = \begin{cases} 0.5 & step < 200 \\ 0.1 & step >= 200 \end{cases} \tag{6}$$

2.3.3 引入额外信息

对于基因变异的部分，尝试引入额外信息辅助变异。对于每次变异，再邻域内枚举20个相邻序列，并根据估价函数(1)选择距离最小的变异作为基因变异的结果。

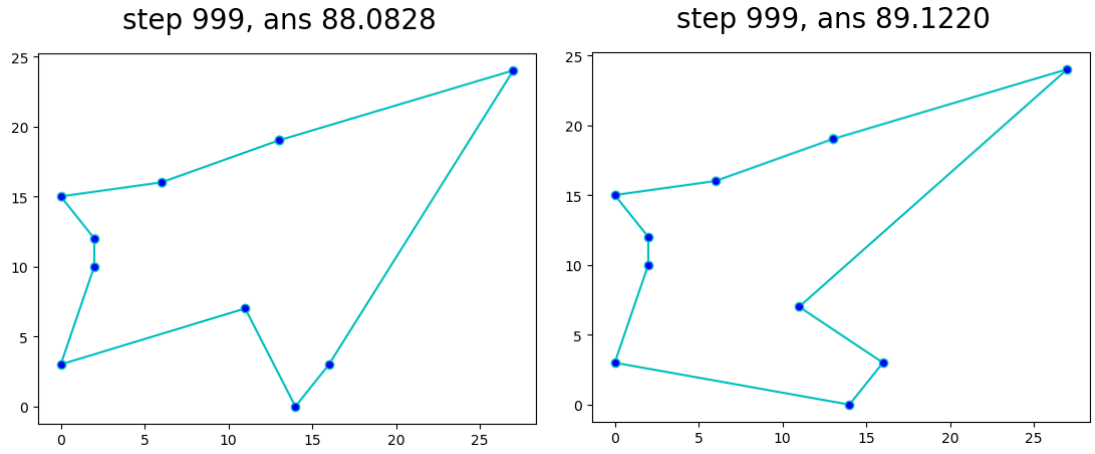
3 实验结果与分析

3.1 数据集与相关说明

本次实验使用TSPLIB[1]中的berlin52问题，该问题采用德国柏林市的地图，包含52个节点。
处于公平比较的原子，本次实验中所有方法都迭代1000次，并且固定随机种子为666072。不过因为绘图的时候涉及到运行先后的问题，实际结果还是不能保证稳定，因此绘图结果和表格结果可能会存在一些差异。对于局部搜索算法和模拟退火算法，每次搜索在邻域的搜索次数为100次。对于遗传算法，种群的大小为500，发生交配的概率为0.5，发生变异的概率见公式(6)。

3.2 准确性测试

首先我设计了一组小数据验证算法的准确性。在随机生成10点图中，三种算法都能够跑到至少接近最优解。以下时测试数据的结果。左边为遗传算法和模拟退火算法的结果，右边为局部搜索算法的结果。其中遗传算法和模拟退火算法跑出了最优解，局部搜索算法则没有跑出最优解，陷入了局部最优解中。



3.3 不同的邻域

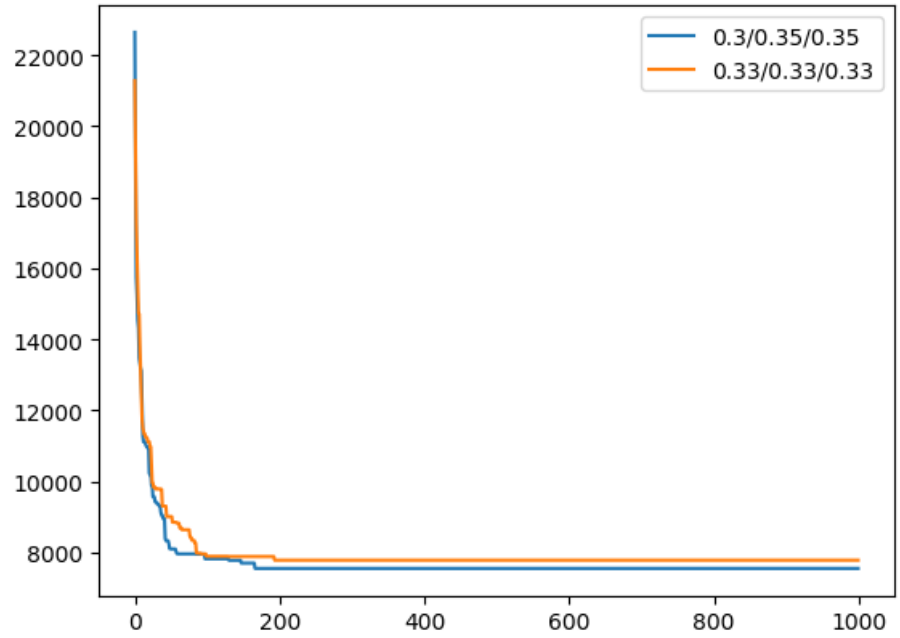
首先根据2.1.2节尝试使用不同的邻域策略，并统计局部搜索得到的汉密尔顿回路的长度。另外，如果使用多种策略，每种策略的使用概率是相同的。如下表所示，将二点邻域和三点邻域结合起来的效果最好。

邻域策略	结果
二点邻域 (1)	12018
二点邻域 (2)	8061
三点邻域 (1)	8131
三点邻域 (2)	8083
二点邻域	8867
三点邻域	8520
二点邻域+三点邻域	7661

因为二点邻域(1)方法的效果一般，因此我尝试将其删去，并将剩余三种方法出现的概率平均化（各三分之一）。我尝试微调一下概率分布，结果却发现产生的汉密尔顿回路长度大相径庭。参数稍微一调，最终的结果竟然差了500多，第二行的结果已经接近全局最优解了。由此可见，局部搜索算法的实验结果并不稳定，根据不同的起点和参数设定，有时候能够到达最优解，有时候则会受困于局部最优解中。

邻域策略	结果
0.33/0.33/0.33	8160
0.3/0.35/0.35	7544

下图为两种参数策略下最优值根据搜索步数的变化情况。可以看到使用局部搜索最短汉密尔顿回路基本在200步之后就不再变化。

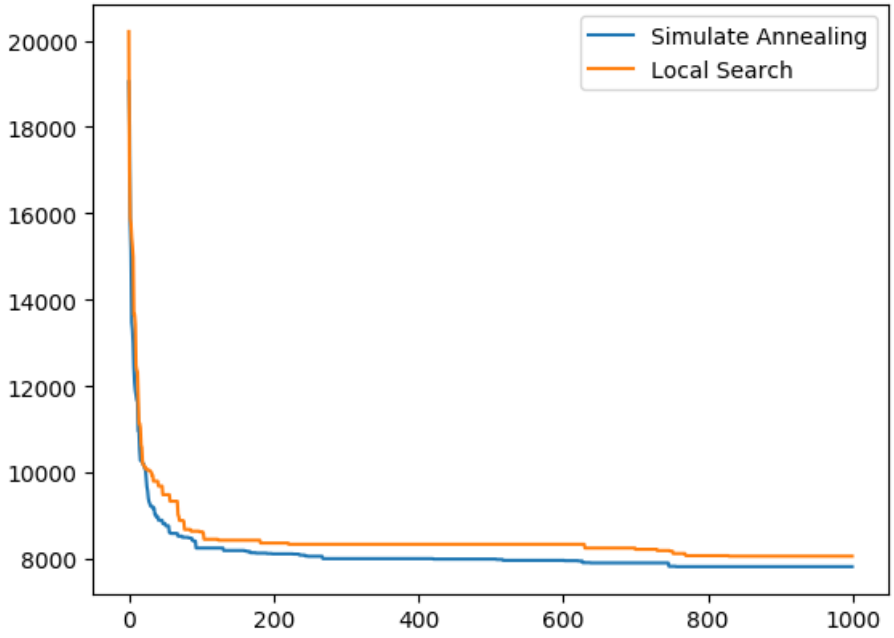


3.4 模拟退火机制的作用

这里设置初始温度1000，尝试在多种邻域策略下使用模拟退火的策略，并和局部搜索法进行对比。可以看到使用模拟退火策略后算法虽然不一定能跑到最优解，但基本上都可以稳定地优化到一个比较优秀的解中，不会受初始值和参数的影响。

邻域策略	局部搜索	模拟退火
0.25/0.25/0.25/0.25	7661	7736
0.33/0.33/0.33	8061	7783
0.3/0.35/0.35	7544	7810

采用 0.33/0.33/0.33 这种邻域策略，分别使用局部搜索和模拟退火算法。下图为两种策略下最优值根据搜索步数的变化情况，通过对比可以看到模拟退火策略的优越性。



3.5 遗传算法的问题与解决方案

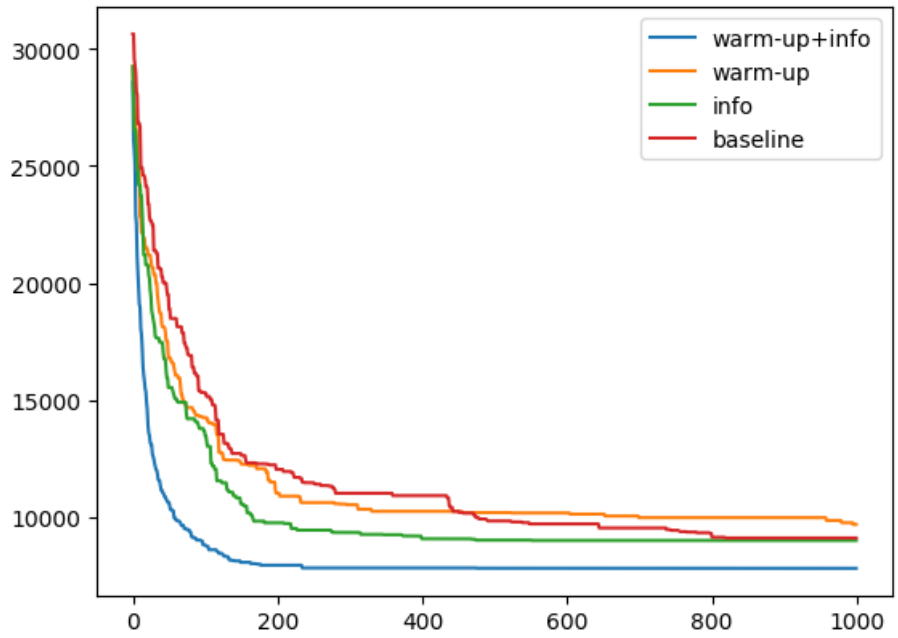
我尝试使用遗传算法进行实验，并使用二点邻域和三点邻域平均测策略进行变异。如3.1节，设置种群的大小为500，交配概率为0.5，变异概率为0.1进行实验。如下表所示，使用遗传算法的实验结果并不理想，路径的长度居高不下，最短路径只有9291，和之前基于局部搜索算法的结果相去甚远。

出现这种情况的主要原因是，遗传算法中染色体是随机初始化产生的，且无引导地进行交配变异，唯一一次有外部信息引导的部分是在选择算子。因此，我认为如果增加一些外部信息引导，遗传算法应该能更好地发挥作用。

我尝试了热身训练和引入外部信息两种策略，如下表所示引入了这两种策略后模型的效果也好了很多。使用热身训练并引入外部信息的情况下，遗传算法的性能可以和使用局部搜索算法的性能相当。

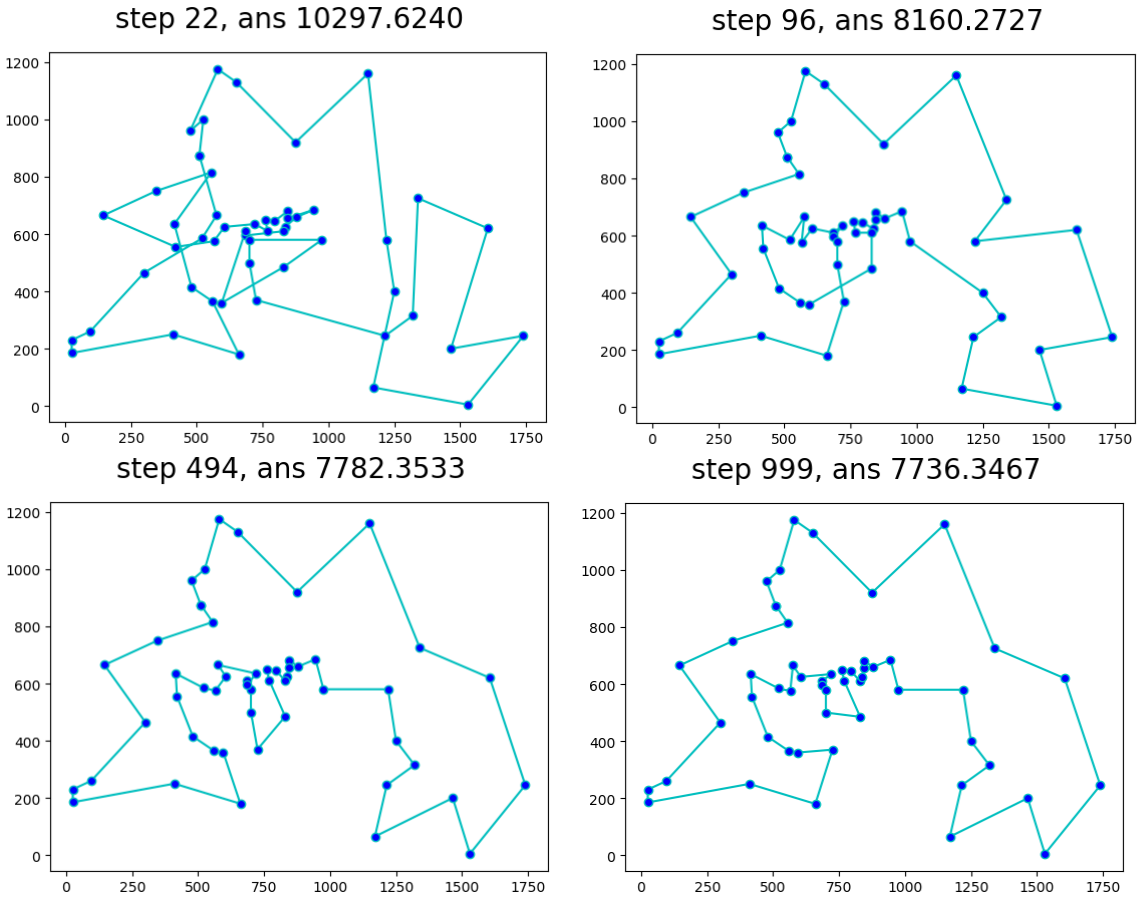
热身训练	外部信息	结果
N	N	9291
Y	N	8329
N	Y	8436
Y	Y	7799

以下为使用几种策略时最优值根据搜索步数的变化情况。可见几种优化策略的作用。



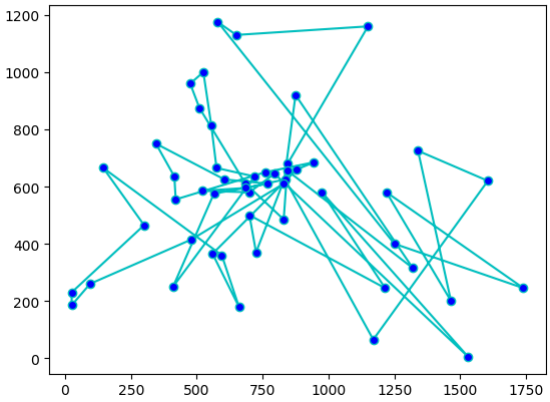
3.6 结果可视化

以下为使用模拟退火算法，采用 0.25/0.25/0.25/0.25 这种邻域策略下最优解随时间变化的情况。可以看到在模型刚开始运行的时候最优解非常混乱，交叉的情况很严重。随着算法的运行，交叉逐渐减少，路径也逐渐成型。

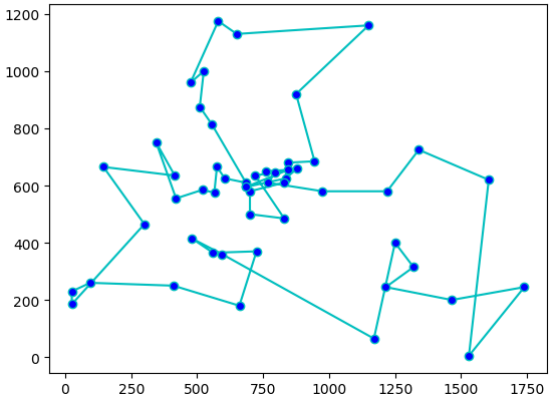


以下为带优化策略的遗传算法进行可视化。

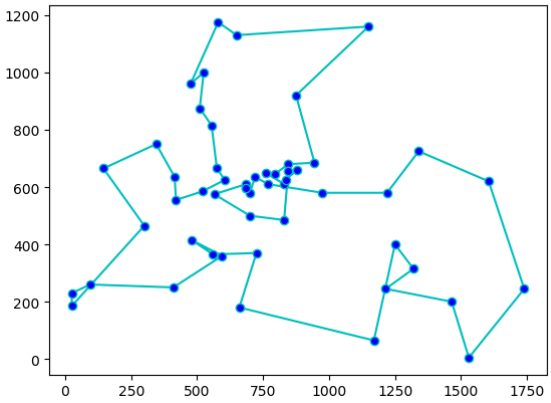
step 11, ans 18056.5090



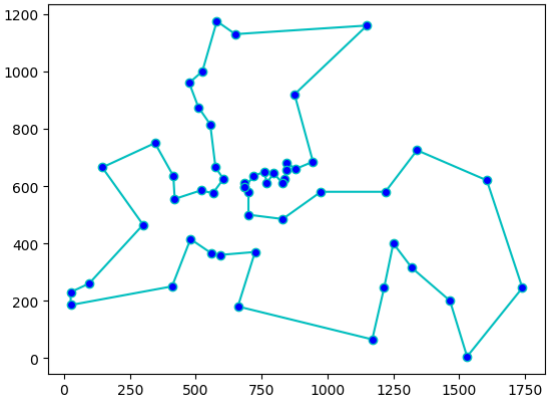
step 60, ans 9889.9360



step 103, ans 8737.9427

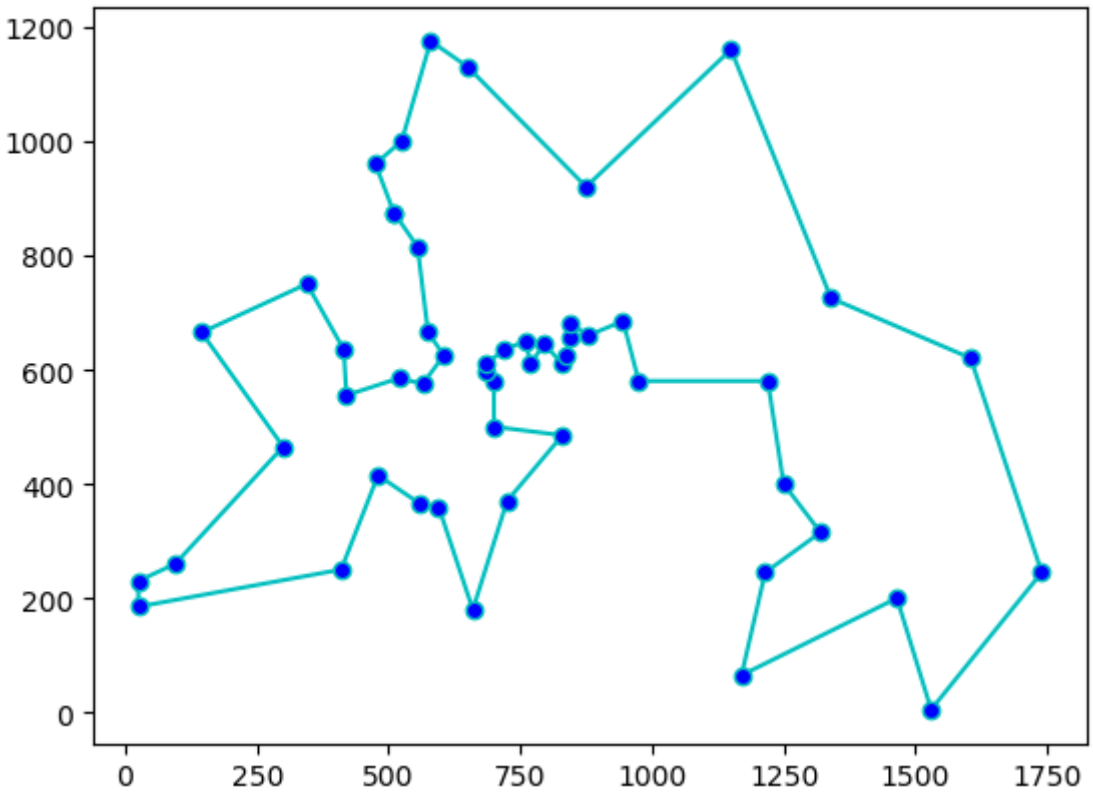


step 999, ans 7799.7992



最后放出一张最优解的可视化结果。

step 999, ans 7544.3659



4 结论

本文尝试使用局部搜索算法，模拟退火算法，遗传算法等多种近似算法解决TSP问题。根据以上实验结果可以发现，使用基于局部搜索的算法可以很好地解决TSP问题，在berlin52问题上的结果非常饥接近全局最优解。简单的局部搜索对参数和初始值的依赖程度较高，很容易陷入局部最优解。而基于模拟退火策略的局部搜索则更加稳定，能够跳出局部最优解。简单的遗传算法并不适合解决TSP问题，为此我们引入了热身训练的机制，并且在变异的时候引入了外部信息，即根据估价函数多次变异选择最优结果。试验结果表明使用经过优化的遗传算法可以取得和局部搜索算法类似的结果。

参考文献

[1] TSPLIB http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html