高级运筹学课程作业

旅行商问题——多方法对比

1. 问题定义

假设地图上有 20 个城市，其坐标如下，旅行商问题需要求解的是一条不重复地经过所有城市的最短路径，本次大作业采取最小插入代价法，蚁群优化和模拟退火算法对规模为 20 以及 100 的旅行商问题进行求解。

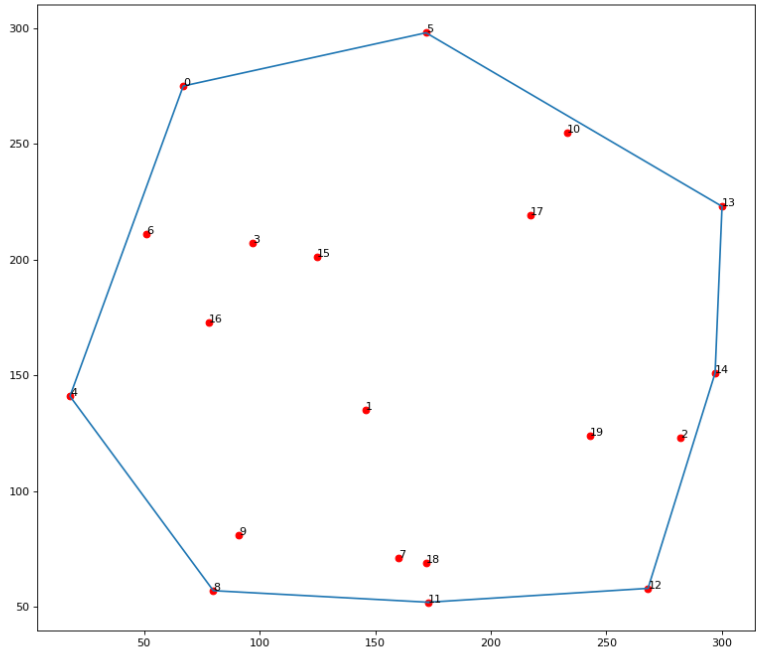


图 1. 30 TSP 问题城市分布

2. 模型建立

为了简化模型，不考虑其它过多的因素影响。 在建模时忽略天气、温度等等外部条件。则目标函数为：

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} d_{ij}$$

其中 x_{ij} 表示是否从 i 城市前往 j 城市，0 表示否定，1 表示肯定。d_{ij} 表示城市 i,j 之间的距离。由于并未采用线性规划的方法解决，对于不同解法而言，约束条件的表现形式并不一样，因此不在此处体现。

4. 构造法

第一步：生成一个城市点集的凸包，以凸包上的点作为临时最优解；

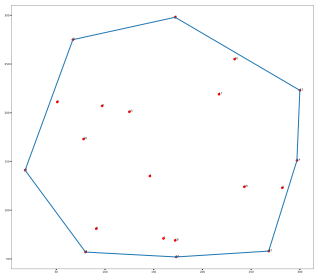
第二步：随机选出不在凸包上的点，并选取最小的代价的间隔进行添加，并纳入临时最优解；

第三步：重复第二步直到所有点被加入临时最优解

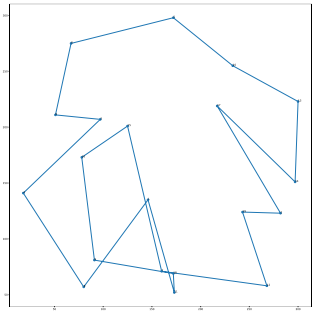
第四步：遍历查询是否有交叉连接(邻域搜索)，并进行替换

第五步：重复第四步直至没有交叉连接

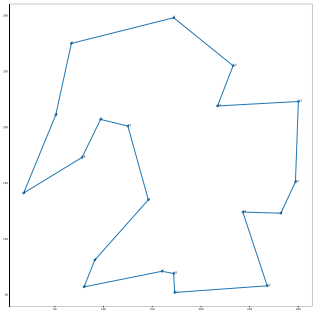
第六步：输出最优序列



第一步



第三步



最优序列

图 2. 构造法最终解

```
final res is: [[173 172  91  78 125 160 268 243 282 217 297 300 233 172  67  51  97  18
   80 146 173]
 [ 52  69  81 173 201  71  58 124 123 219 151 223 255 298 275 211 207 141
   57 135  52]
 [ 11  18   9  16  15   7  12  19   2  17  14  13  10   5   0   6   3   4
    8   1  11]]
target val:  1182.7499644463396
time cost: 617.0055866241455 ms
```

图 3. 构造法结果

5. 方法二：蚁群算法

蚁群算法（ Ant Colony Optimization， ACO）是根据蚂蚁发现路径的行为的而发明的用于寻求优化路径的机率型算法，由 Marco Dorigo 于 1992 年在他的博士论文中提出。蚁群算法是一种模拟进化算法，具有包括较强的鲁棒性在内的许多优良性质，在本质上是一种并行的分布式算法，容易实现，可以较好地求解 TSP 问题。用蚁群算法来求解 TSP 问题主要包括以下几个步骤：

第一步：程序初始化，指定蚂蚁数目，迭代次数等，并为每只蚂蚁随机分配一个起始点

第二步：通过轮盘法为每只蚂蚁选择下一个要到的路标， 并将上一个节点加入本次迭代中该蚂蚁已走过的路标中(实现约束条件)，当所有路标都访问完之后，比较本次迭代中所有蚂蚁的路径成本，记录最佳路径

第三步：更新信息素矩阵。根据每一条蚂蚁爬过的路线，更新每一段路线的信息素浓度。 信息素浓度越高的路线，蚂蚁经过这条路线的可能性越大。且信息素会随着时间减少，所以路径越短，信息素残留的就越多

第四步：检查终止条件，若没达到终止条件则进入下一次迭代。 重复步骤 1~3。

第五步：输出最优序列

```
最佳路径
[ 6.  4. 16.  3. 15.  1.  9.  8.  7. 18. 11. 12. 19.  2. 14. 13. 17. 10.
  5.  0.]
最短距离: 1175
time cost: 4696.825981140137 ms
```

图 4. 蚁群算法结果

6. 方法三：模拟退火算法

模拟退火（Simulated Annealing， SA）算法是求解 TSP 问题的有效方法之一，容易编程实现，求解速度快。模拟退火是一种全局优化算法，加入了随机状态模型，使算法以概率选择的方式逼近最优状态， 而不会像贪心算法一样容易陷入局部最优解。

模拟退火解决 TSP 的思路：

第一步：随机产生一条路径 P(i),作为初始值。

第二步：产生一条新的遍历路径 P(i+1)，计算路径 P(i+1)的长度 L(P(i+1))若 L(P(i+1)) < L(P(i))，则接受 P(i+1)为新的路径，否则以一定的概率接受 P(i+1) ,此步骤模拟退火过程，接受 P(I+1)之后，概率（即温度）会降低，同时越到最后降低速率越慢。

第三步：重复步骤 1）， 2） 直到满足退出条件， 输出最优值。

退火过程中产生新的遍历路径的方法有很多，下面列举其中 2 种：

（ 1） 随机选择 2 个节点，交换路径中的这 2 个节点的顺序。

（ 2） 随机选择 3 个节点 m， n， k，然后将节点 m 与 n 间的节点移位到节点 k 后面。

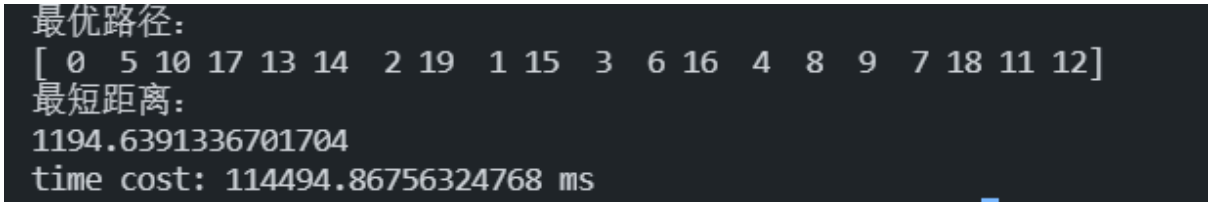
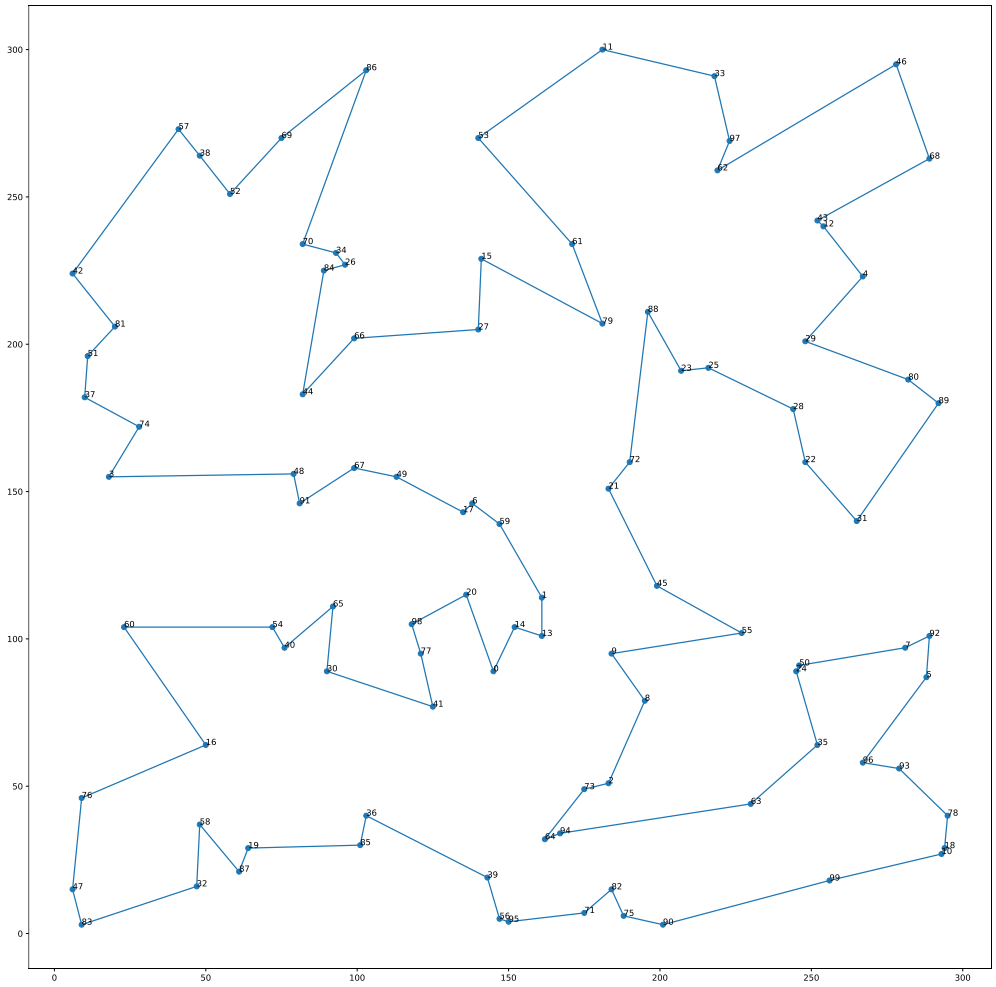


图 5. 模拟退火算法结果

7. 大规模计算效率对比

从规模为 20 的 TSP 问题求解的结果来看，构造法兼具求解准确度和计算速度，蚁群算法速度较慢，而模拟退火算法的计算准确度和计算效率都最差。考虑到启发式算法有着较为考究的调参过程，因此计算效率和准确度仅作为定性参考。

群体优化算法在求解大规模问题时，计算复杂度一般而言不会以高阶的速度增长，因此本节对比一下 100 个城市的 TSP 问题求解效率。



```
final res is: [[  9  61   9  76  23  72 101 118 121 223 289 267 256 279 184 150 167  92
  196  79  81  20  28  99  89  75  82 282 181  99 190 188 147 175 162 175
  230 113  11  58 140 219 171  82 125  48  47 147 201 227 195 199 252 246
  245 248  48 218 252 265 244 184 143 183 103  10  90  93 138  96 140 141
  161 152 136 135 216 183 207 248 161  18  50  64 145 293 294 288 254 281
  295 292 267 289 278 181 103  41   6   6   9]
 [  3  21  46  97 104 104  30 105  95 269 101  58  18  56  15   4  34 111
  211 156 146 206 172 158 225 270 234 188 207 202 160   6 139  49  32   7
   44 155 196 251 270 259 234 183  77  37  16   5   3 102  79 118  64  91
   89 201 264 291 242 140 178  95  19  51  40 182  89 231 146 227 205 229
  101 104 115 143 192 151 191 160 114 155  64  29  89  27  29  87 240  97
   40 180 223 263 295 300 293 273 224  15   3]
 [ 83  87  76  40  60  54  85  98  77  97  92  96  99  93  82  95  94  65
   88  48  91  81  74  67  84  69  70  80  79  66  72  75  59  73  64  71
   63  49  51  52  53  62  61  44  41  58  32  56  90  55   8  45  35  50
   24  29  38  33  43  31  28   9  39   2  36  37  30  34   6  26  27  15
   13  14  20  17  25  21  23  22   1   3  16  19   0  10  18   5  12   7
   78  89   4  68  46  11  86  57  42  47  83]]
target val:  2554.8845334621483
time cost: 7162.375211715698 ms
```

图 6. 构造法结果(city_num=100)

对于构造法而言，$T_{100}/T_{20}=11.61$

```
[64.  5. 54. 65. 44. 43. 75. 67.  8. 62. 96. 78.  4. 50. 85. 91. 83. 89.
 73.  7. 74. 27. 31. 45. 63. 34. 30. 36. 76. 29. 97. 22. 10.  9. 35. 19.
  2. 71. 49.  0. 16. 48. 32. 70. 39. 14. 23. 82. 57. 11. 38. 55. 52. 88.
 46. 33. 41. 99. 21. 59. 95. 81. 13. 93. 79.  6. 80. 69. 37. 51. 61. 40.
 92. 66. 90. 15. 86.  3. 87. 42. 58. 12. 72. 98. 56. 18. 84. 26. 77.  1.
 60. 47. 20. 68. 17. 94. 28. 25. 24. 53.]
最短距离: 2615
time cost: 88150.1054763794 ms
```

图 7. 蚁群算法结果(city_num=100)

对于蚁群算法而言$T_{100}/T_{20}=18.77$

```
最优路径:
[ 0 95 24  4 89 15 49 96 82 91 64 80  7 52 51 66 19 53  9 90 79 55 21 44
 60 31 45 54 35 92 41 27 83 58 23 50 36 87 14 71  3 48 32 57 16 28 10 18
  8 62 65 29 69 42 63 85 84 74  1 70 34 12 38 86  2 81 11 33 20 59 68 25
 39 78 47 77 37 22 75 99 17  5  6 93 72 88 43 98 13 73 97 40 26 61 56 46
 30 76 94 67]
最短距离:
2583.93218413081
time cost: 307437.956571579 ms
```

图 8. 模拟退火结果(city_num=100)

对于蚁群算法而言$T_{100}/T_{20}=2.69$

由于是 100 城市的 TSP 使用的是随机数据，最优解不足以说明问题，但是能够看到，模拟退火算法对于问题规模的敏感度(竟然)远低于其余两种算法。

具体原因在于其并非多个体的群体优化算法，因此对于问题规模不敏感，其耗时可能大部分在于跳出最优解，或者收敛速度较慢，亦即如果有合适的降温曲线、终止准则，模拟退火算法的计算效率能够得到可观的提升。

对于启发式算法而言，当问题规模增大，插入过程的遍历会导致计算量与规模相关，可以考虑不进行遍历，进抽取部分连接进行插入代价比较。