基于对称 TSP 问题的研究

20184484 胥卜凡 20184439 颜扬升 20184517 董修良

20184544 王有为 20184417 于永勋

摘　要　旅行商问题（简称 TSP）是一个著名的 NP-Hard 问题，也是离散优化的一个经典的重要问题，对其相关求解算法的研究非常重要。本文在介绍了 TSP 问题本身相关的问题后，又详细讨论了求解 TSP 问题的动态规划方法、改良圈算法、二交换算法、模拟退火算法、蚁群算法、遗传算法，并通过整合各种优化方式，对遗传算法进行了少量优化。针对测试库中的的改良圈算法、模拟退火算法、蚁群算法、遗传算法。本文使用 MATLAB 软件实现这些算法，并对这些算法的运行时间和解进行比较分析等研究。结果表明在任何清空下改良圈算法的性能都最差，其余算法在小型 TSP 问题下差别不大，在较大型 TSP 问题下，模拟退火所用时间最短，蚁群算法所用时间最长，遗传算法的解最好。实验结果证明，改进后的遗传算法效果不明显，在解的大小上改善 1%-3%，但时间提高了 10%。

关键词　对称 TSP 问题；算法研究；近似算法;模拟退火;遗传算法;蚁群算法

Study based on symmetric TSP problem

Xu Bu-Fan　Yan Yang-Sheng　Dong Xiu-Liang　Wang You-Wei　Yu Yong-Xun

Abstract　Travelling salesman problem (TSP) is both a well-known problem as a NP-Hard problem and a classical problem in the field of discrete-optimization. This paper first introduce something about itself, then discusses many ways such as dynamic programming, 2-opt, exchange algorithm, simulated annealing algorithm, ant colony optimization and genetic algorithm in detail. After that, this paper proposes an improved genetic algorithm by the means of integrating many papers' methods.Finaly, this paper .Experiments show that Improved cycle algorithm perFORms worst, and when it comes to small TSP problems, those algorithms perFORms familiar. However, when solving big TSP problems, simulated annealing algorithm solves quickest as ant colony optimization takes longest, and YICHUAN makes the best solution. Experiments also show that the improved algorithm makes approximately 3%~5% improvement FOR the last length but time takes 10% more.

Key words　Symmetric TSP problem; Algorithm research; approximate algorithm; simulated annealing; genetic algorithm; ant colony algorithm

1 引言

1.1 TSP 问题

TSP 问题，即一个商人想要去 n 个城市推销商品，每两个城市 i 和 j 之间的的距离为 d_{ji}，如何选择一条道路使得商人能够每个城市仅走一遍后回到起点且保证所求的路径最短。其中，当对所有的城市 i 和 j 都有 $d_{ij}=d_{ji}$，则称为对称 TSP 问题，否则称为非对称 TSP 问题，本文主要讨论对称 TSP 的

问题。

对NP问题,如果采用标准的穷举算法,复杂度会达到O(n!)的复杂度。所以目前学术界对这类问题经常使用近似算法来求解。但这种算法,大量的启发式的搜索算法不一定能求出最优解,往往会陷入局部收敛,如何跳出局部收敛得到最优解也是现在相关算法改进的方向。

为方便后序的讨论,我们可以将TSP问题转化为数学模型:

城市$V=\{v_1, v_2, \cdots, v_n\}$的一个访问顺序为$T=(t_1, t_2, \cdots, t_n)$,其中$t_i \in V$,且$t_{n+1}=t_1$,则TSP问题要求的就是$\min \sum_{T\in\Omega}^{n} d_{t_i t_{i+1}}$,其中$\Omega$是所有的不重复排列中可能的回路,*n+1*代表的点v_1。[5]

1.2 研究意义

作为NP问题的代表性问题,TSP问题具有典型的易于描述却难以大规模处理的特性[4],是在较多领域内出点的多种复杂问题的集中概括和简化形式。对它进行相关算法的设计、优化、讨论无论是对类似的图问题的拓展,还是对相关的NP-Hard问题算法的设计与实现,都具有比较广泛的现实意义。

1.3 本文的主要工作

本文针对TSP问题,首先介绍了目前学术界大多数的TSP相关算法,包括GA算法,模拟退火算法,动态规划算法,改良圈算法,二交换法,三交换法等,并通过已有的文献对这些算法进行了比较分析,并对遗传算法实现一些少量的改进等,最后通过本文相关算法进行了比较分析。

具体的研究内容如下:

(1)分析了各算法的基本原理以及解决TSP问题的方法、流程。

(2)对大多数算法进行了分析,指出各个算法的优缺点。

(3)针对遗传算法通过整合大量的优化方式,进行少量的改进。

(4)利用matlab等软件,使用TSPLIP测试集,进行算法结构改进方面的实验进行实验结果的比较分析。

2 相关概念与工作

2.1 TSP相关算法概述

目前对TSP的大部分研究都是在非确定性的相关算法下进行的,但我们也应该注意到传统的确定性算法,因为对传统的算法的改进也能或多或少的运用。因此我们根据大部分TSP相关的论文,进行了整体的优劣性的相关比较,如下图所示。

其中传统的确定性算法在总体分析上具有时间复杂性大、空间复杂性大的特点,这主要是由于问题本身就是一个NP完全问题,而现代流行的近似算法则具有算法复杂性小的特点,很快就能得到较优的解,下面本文分别对上述的算法进行介绍,并进行比较分析。

总体来说,传统的算法在总体分析上具有时间复杂性大、空间复杂性大的特点,这主要是由于问题本身就是一个NP完全问题,而现代流行的只能算法则具有算法复杂性小的特点,很快就能得到较优的解。

2.2 动态规划算法

动态规划DM(Dynamic Prog ramming,简称DM)是运筹学的一个分支,是针对多阶段决策过程的优化问题[7]而被提出的,他能把多阶段过程转化为一系列的单阶段的子问题,逐个进行求解,即将待求解问题分解成若干子问题,然后从这些子问题中的解得到原问题的解。

针对TSP问题,假设我们记S为集合

{2，3，..}的子集，$k \in \{2,3,...,n\}$，$C(k,S)$为从 k 出发遍历 S 中的节点并终止于节点 1 的最短距离。当$|S| \geq 1$时，根据动态规划法的最优性原则，我们可以得到 TSP 问题的动态规划方程：

$$C(k,S) = \underset{j \in S,\ k \cap S = \phi}{Min} \{d_{kj} + C(j, S - \{j\})\} \quad (0.1)$$

由于最后TSP问题最终形成的是一条闭合回路，故我们可以任意设定旅行的起始位置，不妨就设定为起点 1，那么我们可以得到最短路径：

$$Min\{C_{1k} + C(k,V)\}, \text{其中 } k \in V \quad (0.2)$$

显然，该算法属于递归算法的范畴，从递归方程中我们可以得到该算法的时间复杂度为$O(n^2 2^n)$，空间复杂度为$O(n2^n)$，是一个典型的 NP 问题的算法所具有的算法复杂度。

2.3 改良圈算法

哈密顿图（哈密尔顿图）（英语：Hamiltonian path，或 Traceable path）是一个无向图，由天文学家哈密顿提出，由指定的起点前往指定的终点，途中经过所有其他节点且只经过一次。在图论中是指含有哈密顿回路的图，闭合的哈密顿路径称作哈密顿回路（Hamiltonian cycle），含有图中所有顶点的路径称作哈密顿路径。

从图中的任意一点出发，路途中经过图中每一个结点当且仅当一次，则成为哈密顿回路。（我的理解就是从一点出发经过所有点在回到原始位置画一个圈）

Hamilton 圈满足两个条件:①封闭的环②是一个连通图，且图中任意两点可达

由于 TSP 问题的母图是竞赛图，所以 TSP 问题可以转化为求总路径长度最短的那个 Hamilton 圈

算法首先求得一个 Hamilton 圈，然后修改圈得到具有较小权的另一个 Hamilton 圈，直至无法改进则停止。[9]

2.4 交换算法

交换算法输入密码学的范畴，公开密钥交换算法在网络通信中扮演着重要的作用。例如，密钥交换算法是一些最常用的密码学协议中的重要组成部件(SSL， IPsec， SSH 等等)，这些协议保证了网络通信以及电子商务的安全快速发展。因此，密钥交换算法自从诞生以来，一直是现代密码学家的关注重点。

针对 TSP 问题，比较常见的两种算法是二变换法和三变换法。其中二变换法的操作是，任选当前路径的序号(u,v)，交换 u 和 v 之间的访问顺序，即若交换前路径为

$$s = (c_1,...,c_u,...,c_v,...,c_n) \quad (0.3)$$

则交换后的路

$$s = (c_1,...,c_{u-1},c_v,c_{v-1}...,c_{u+1},c_u,c_{v+1}...,c_n) \quad (0.4)$$

三交换法同理，任选三点，将两个点中间的路径与另一个点之间交换即可。

2.5 模拟退火算法

与传统的确定性算法想办法消除搜索消耗的影响不同，大部分的人工智能算法是在搜索的框架下进行的，他们不是完美的求解算法，都只能能在一定精确范围内求解，但时间复杂度很低，效率很高，因此得到了广泛的应用。

模拟退火算法（Simulate Annealing Arithmetic，SAA）是一种通用概率演算法，用来在一个大的搜寻空间内找寻命题的最优解。它是基于 Monte-Carlo 迭代求解策略的一种随机寻优算法。

SA 算法是在爬山法[见附录]的基础上进行的，爬山法是一种完完全全的贪心算法，难以避免总会收敛到局部最优解，SA 算法则在搜索到局部最优解后，会以一定的概率接受继续移动，这就能跳出局部最优继续搜索。SA 算法的名称得益于上述概率的计算公式，即使用了

$$P = \exp(-\Delta c'/T) \quad (0.5)$$

此外，为了能够产生新解，模拟退火算法需要用到新解产生的算法，比较常见的两种算法是二变换法和三变换法。

得益于爬山法的贪心性质，模拟退火算法具有较强的局部寻优能力且不容易在搜索的过程中陷入局部最优解。但是算法仍然不便于把握全局最优解，且参数温度 T 的初始值，退火的速度，温度管理方面都难以进行控制。[10]

2.6 蚁群算法

蚁群算法(Ant Colony O ptimization,简称 ACO)由意大利学者 Dorigo M 等在 1996 年首先提出 15,16]。该算法通过模拟蚂蚁的觅食行为，蚂蚁觅食的时候会在所走过的路径.上留下信息激素,同时信息激素会随时间而挥发。一条路径走过的蚂蚁越多，留下的信息激素越多;反过来,信息激素浓度高的路径上会吸引更多的蚂蚁。通过这种正向反馈,最终将找到一条最优路径。为了避免蚂蚁两次走上同条路径(非法路径)，为每个蚂蚁设置一个禁忌表以记录它已走过的路径。

蚁群算法与其他算法相比是一种具有自适应、自组织的方法，他利用了正反馈节的思想，促使整个系统逐渐向最优解进化，且参数少，易于调整，往往可以和其他算法组合到一起。[12]

2.7 遗传算法

遗传算法(Genetic Algorithm,GA)是一种通过模范自然进化过程搜索最优解的方法。其主要特点是直接对结构对象进行操作，不存在求导和函数连续性的限定，不需要确定规则就能自动获取与指导优化搜索空间，自适应的调整搜索方向。[8]

主要的基本算子包括选择算子、交叉算子和变异算子。传统的遗传算法计算过程如下：编码，初始化，计算适应度，轮盘赌挑选下一代，交叉，变异形成新种群，继续迭代。

2.8 郭涛算法

武汉大学的郭涛博士在 1999 年提出的一种具有交叉、变异特征的 In-Over 算子，是目前最为有效的解决TSP问题的方法之一。

由于郭涛算法是在遗传算法的基础上改进的，故本文仅对其改进的 In-Over 算子进行简单的介绍：

(1)变异算子：

在父体 S 中随机选择两个城市 C,C'，对 C 的下一个城市与 C' 间的所有城市(包括 C')进行倒位操作。

(2)杂交算子：

在父体 S 中随机选择一个城市 C，另取一父体 S'，在 S' 中指定 C 的下一个城市为 C'。对 S 中 C 的下一个城市与 C' 间的城市实施倒位操作，若 S 中 C 与 C' 相邻，则不进行倒位操作。[9]

上述介绍中的倒位操作，即若在 $(1,2,3,4,5,6,7,8)$ 选择 2，6 城市作为 C 与 C'，则倒位后结果为 $(1,2,6,5,4,3,7,8)$，该算法的具体流程如下:

(1)初始化种群 P，变异率 p。

(2)对于每个个体 $S_i \in P$,令 $S'=S_i$

(3)从 S_i 中随机选取一个城市 C

(4)产生一个 0~1 的随机数 rand

(5)判断 rand≤ p 是否成立,成立则转向(6)，否则转向(7)

(6)从 S' 剩下的城市中随机选择一个城市 C'

(7)从种群 P 中随机选择一个个体 X，令 C'=在 X 中 C 城市的后一个城市

(8)判断在个体 S' 中城市 C 是否与 C' 相邻，相邻则转向(11)，否则转向(9)

(9)把 S' 从城市 C 到 C' 中的部分倒位

(10)令$C=C'$，转向(4)

(11)判断新的个体S'的适应度是否优于个体S_i，成立则令$S_i=S'$

(12)判断是否满足临界条件，满足则结束；否则转向(2)

在经过大量的测试集测试后，可以看出In-Over算子实现简单，执行效率高，算子具有高度的杂交与变异的特征。由于算法不停地对基因片段进行逆转操作，在算法的前期不容易使路径中存在交叉的边，即防止覆盖问题的发生。对基因片段不断进行逆转使其能够增加种群的多样性，所以算法的早期收敛快。但当算法到达后期时，所有的染色体表示的解的路径基本都没有交叉了，再进行盲目的逆转是极其浪费时间的。所以该算法仍然具有容易陷入局部最优值的特点。

3 算法设计

3.1 模拟退火算法

3.1.1 算法设计思想

模拟退火算法是一种基于蒙特卡洛思想设计实现的近似求解最优化算法，所以算法设计需要考虑迭代次数，每次迭代时通过各种方法，如二交换法、三交换法等来进行目标路径的更新。针对对称TSP问题，目标函数设置为路径总长度，在进行判断时，如果新一轮的计算结果更前一轮之结果更小，那么我们就接受它，否则就以一个概率来拒绝或接受它，而这个拒绝的概率会随着温度的降低（也即是迭代次数的增加）而变大（也就是接受的概率会越来越小）。最后，由于模拟退火是在贪婪算法的搜索过程引入了随机因素，所以可以先用蒙特卡洛等方法得到一个较好的初始解来进行搜索。

算法的核心思想是近似蒙特卡洛的穷举，解空间属于排列树，在搜索解空间的过程中通过应用热力学的概率计算来跳出局部收敛。

3.1.2 伪代码

```
初始化参数；
利用蒙特卡洛方法得到一个较好的初始解；
FOR k=1:L
   C=随机方式产生新解
   C=sort(c);c1=c(1);c2=c(2);
   df=二交换法等产生新解
   IF(df<0)%接受准则
   更新 path
   ELSEIF exp(-df/T)>rand%以一定概率接受
   更新 path
   END
   T=T*at;
   IF(T<误差限界 e)
        Break;
END
```

3.2 改良圈算法

3.2.1 算法设计思想

算法首先通过随机生成等方法求得一个哈密尔顿圈，之后通过将节点之间进行逆序操作来构建新的改良圈，如果改良后的圈的总路径更小就更新目标圈，直到无法改进或者达到跳出的条件结束算法。

算法的核心思想是近似蒙特卡洛的穷举，解空间属于排列树，但通过判断局部收敛能够跳出穷举。

3.2.2 伪代码

```
C=通过蒙特卡洛等方法得到初始圈
FOR k=1:L(迭代次数)
     Flag=0;退出标志
     FOR i=1:L-2
          FOR j=i+2:L
               IF 改良成功
                    更新
               END
          END
     END
     IF flag==0;改良收敛
          BREAK
     END
END
```

3.3 动态规划算法

3.3.1 算法设计思想

将所求的最优路径划分为子问题，从初始节点出发，列出该节点之外排列中的各种可能的解，按顺序求解子问题，子问题的解来判断该点是否保留在最优路径中，即选择去掉该点后总路径长度最小的那个点作为该节点的下一个点。

3.3.2 伪代码

```
FOR j=1:V-1%按列遍历不同集合
    FOR i=1:n%遍历城市
        X=判断 i-1 在不在 V（j,:)数组里
        IF 不在（否则会经过两次）
            FOR k=0:n%试探下一步为集合中的任何一个
                Index=对去掉当前已走过城市后，路径方案的下标检测
                IF(V(i,k)!=0&&(c(i,V(j,k))+d(V(j,k),index))< d(i,j))
                    d(i,j)更新
                END
            END
        END
    END
END
FOR k=0:n%试探下一步为集合中的任何一个
    IF(v(V-1,K)!=0)&&(c(0,V(j,k))+d(V(j,k),index))  < d(0,V-1))
    更新 d(0,v-1)
    END
END
```

3.4 蚁群算法

3.4.1 算法设计思想

每次随机选择点作为初始点出发，通过启发式信息和信息素浓度来选择下一个城市进行路径的求解，在所有蚂蚁走完了所有城市后，通过路径的重叠信息，进行信息素浓度的更新，之后不断进行迭代，得到较优的解。

算法的核心思想是同时进行大量的解搜索，并利用之前的解给下一次迭代提供搜索信息，可以说是优化的蒙特卡洛思想。

3.4.2 伪代码

```
初始化参数;
FOR k=1:L（迭代次数）
    TABLE=随机产生各个蚂蚁的起点城市;
    Index=构建解空间;
    FOR i=1:蚂蚁个数
        FOR j=2:城市个数
            Tabu=禁忌表
            Allow=待访问城市
            FOR n=1:length(allow)
                P(n)=计算转移概率
            END
            轮盘赌选择下一个城市
        END
    END
    FOR n=1:蚂蚁个数
        Length(n)=路径距离
    END
    BL=最优距离
    FOR i=1:蚂蚁个数
        FOR j=1:城市个数-1
            更新信息素
        END
    END
    清空相关表
END
```

3.5 遗传算法

3.5.1 算法设计思想

通过建立多个种群进行选择，交叉，变异的操作迭代获得最优解。其中选择操作可通过轮盘赌的方法选择是否保留当前解集合，交叉操作将两个可行解通过交换等方式进行更新替换，变异操作通过对一个解进行单点交换等操作进行。

算法的核心思想是模拟大自然的自然选择的进化，来获得当前搜索到的解空间的最优解。

3.5.2 伪代码

```
初始化参数，概率列表 P;
FOR k=1:g
    FOR i=1:2:w（种群个数）
        交叉产生染色体
    END
    Table=rand();产生变异的地址
    FOR j=1:NUM(Table)
        对 TABLE 中的染色体进行变异
    END
```

```
    [sloong,ind2]=sort(LENGTH);
    FOR j=1:w
        R=pmax*rand;
        FOR m=1:w
            P(m)概率保留
        END
    END
END
```

3.6 遗传算法的优化

3.6.1 算法的设计思想

如前文所述，遗传算法具有自组织、自适应和自学习性，这使他同时具有能够根据环境变化来自动发现环境的特性和规律的能力，且由于交叉算子的存在，遗传算法能将注意力集中到搜索空间中期望值最高的部分，另一方面，通过变异使种群多样性得到保证，确保种群能够持续进化，使遗传算法能以更大的概率找到全局最优解。但是同时，遗传算法存在收敛速度慢、算法运算量较大的缺点。它既没有用到目标函数的梯度，也无法直接确定解在解空间中的位置，无法直接判断算法是否收敛，当有某个解较优时，有可能会使迭代后的个体与父代群体越来越相似，使非最优解占主导地位而陷入局部极值点。[1]

产生局部收敛的原因一方面包括群体多样性减少的过早，使搜索的空间受到了极大的限制；另一方面是由于传统算法中选择交叉变异的算子不够良好。

针对遗传算法存在的缺点，目前大部分的实现集中在三个方面，一是对遗传算法基本结构的改进，另一个是将遗传算法与模拟退火，神经网络等其他方法进行结合，最后一个是利用遗传算法的本质上的并行性建立并行的遗传算法。[5]

本文采用的方法是将各种优化方法结合到一起，对各个结构进行优化。

3.6.2 改进方法[5]

（1）初始化种群时，使用蒙特卡洛等策略或者采用改良圈算法得到一个较优的解。

（2）在变异和使用混沌序列进行映射。[8]

（3）交叉时使用类似模拟退火的温度概率进行计算。

（4）变异时随机选择多种变异方式的某一种。

4 实验

4.1 实验环境

处理器：Intel(R) Core(TM) i7-8650U CPU @ 1 .90GHz 2.11 GHz

IDE ：MATLABr2018b

4.2 实验结果分析

4.2.1 改良圈算法

参数设置：

无具体参数设置，数据集为 d198.tsp 其他算法数据集使用相同，后序不再进行说明。

实验结果：

表 1 改良圈的结果

类型	时间	距离	最优距离
berlin52	0.018737	16174.78	15968.47
bier127	0.09866	261627.5	257718.5
ch130	0.107877	13416.79	13217.48
ch150	0.152037	14881.52	14720.16
d198	0.247262	32992.44	33108.92
d493	1.375561	76523.84	75944.12
d657	3.237313	1.09E+05	1.09E+05
eil101	0.057108	1404.75	1374.77
eil76	0.034667	1221.305	1211.638
fl417	8.47E-01	25752.32	25751.35
gil262	0.3064	5233.8	5122.9
gr229	0.266225	3559.673	3544.13
gr431	0.9482	4146.159	4131.943
kroA100	0.053486	47036.78	46152.09
kroA150	0.150549	59262.21	57886.62
kroA200	0.269232	64525.25	63877.2
kroB100	0.059159	47752.64	46327.85
kroB150	0.163032	57906.56	57401.4
kroB200	0.230662	64322.31	63271.42
kroC100	0.056626	47653.01	44776.64
kroD100	0.05879	46388.87	46248.7
kroE100	0.05506	48163.51	47181.03
lin105	0.063794	31273.35	30868.89
lin318	0.48102	93703.76	92738.62

p654	2.992822	75093.4	74840.72
pcb442	1.068019	114938.2	112512.8
pr107	0.066396	93608.16	93233.23
pr124	0.066512	93986.48	93318.93
pr136	0.143164	212626.7	209468.5
pr144	0.123907	122731.3	120995.3
pr152	0.155114	157400.6	155106.4
pr299	0.433754	108921.2	108908.4
pr76	0.032569	232001.9	227828.1
rat195	0.241768	5146.548	5145.393
rat575	2.457147	15246.18	15174.52
rat99	0.051934	2762.305	2679.68
rd100	0.070312	17204.42	16680.71
rd400	0.89451	33659.02	33323.07
st70	0.025493	1450.437	1403.932
ts225	0.268291	272905.5	272434.7
tsp225	0.260262	8515.003	8493.402
u159	0.191187	94477.48	92205.07
u574	2.172184	82881.2	82162.32
u724	4.122716	94197.92	93814.86

可视化：

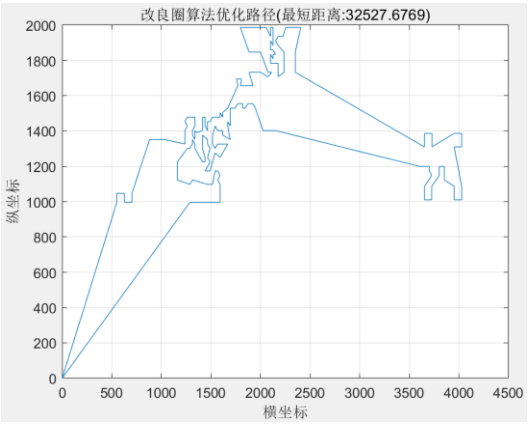

图 1 改良圈可视化

4.2.2 模拟退火算法

参数设置：

终止误差限：0.1^200;

初始温度:1

温度下降指数：0.999

迭代次数：100000

实验结果：

表 2 模拟退火的结果

类型	时间	距离	最优距离
berlin52	0.145147	8465.11	8400.746
bier127	0.131059	130182	129442.8
ch130	0.13954	6705.328	6537.498
ch150	0.150373	7300.676	7270.61
d198	0.147084	16781.51	16657.76
d493	0.192149	53283.3	52378.44
d657	0.21692	98320.01	97479.18
eil101	0.127745	704.0532	695.053
eil76	0.150052	602.5421	600.1821
fl417	0.183065	19167.04	18981.42
gil262	0.172238	2799.972	2770.13
gr229	0.150993	1818.993	1774.716
gr431	0.174207	2663.888	2644.83
kroA100	0.13477	22176.78	21885.09
kroA150	0.154067	28375.33	28286.68
kroA200	0.15589	33327.16	32688.28
kroB100	0.12676	24923.71	24809.7
kroB150	0.161998	28784.02	27970
kroB200	0.147181	34484.72	33981.19
kroC100	0.137048	22848.52	21616
kroD100	0.129223	23229.67	23191.27
kroE100	0.142819	23598.79	22918
lin105	0.133079	15751.15	15497.12
lin318	0.1686	53447.3	53311.72
p654	0.23629	73769.82	68756.32
pcb442	0.180826	77091.42	75433.36
pr107	0.139973	47728.4	46923.6
pr124	0.14459	62089.99	61754.85
pr136	0.137486	105540.8	105134.6
pr144	0.159086	65806.62	60769.87
pr152	0.159034	78659.44	75960.11
pr299	0.158027	60664.99	60546.7
pr76	0.133733	114908	114413.6
rat195	0.16053	2666.519	2658.08
rat575	0.203402	11502.55	11427.81
rat99	0.132421	1357.716	1346.39
rd100	0.133383	8797.541	8670.91
rd400	0.175452	21451.77	21319.88
st70	0.138232	735.7213	729.0573
ts225	0.161985	138628.7	135808.6
tsp225	0.15612	4377.392	4371.009
u159	0.16443	46698.71	44991.29
u574	0.200855	67009.05	66300.36

u724	0.231609	90312.4	90128.9

可视化：

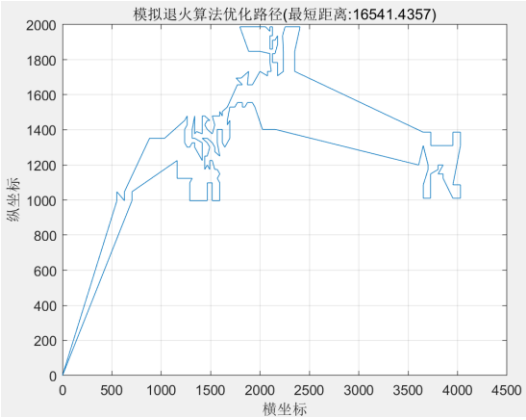

图 2 模拟退火可视化

结果分析：

4.4.3 蚁群算法

参数设置：

蚂蚁数量:31

挥发因子：0.2

最大迭代次数:100

实验结果：

表 3 蚁群的结果

类型	时间	距离	最优距离
berlin52	2.8446	7811.97	7542
bier127	10.3772	125374.2	118282
ch130	11.0422	6452.804	6110
ch150	14.5862	6880.817	6528
d198	21.5234	17549.19	15780
d493	110.121	40472.21	40132
d657	204.854	59204.54	58864
eil101	7.3918	711.0904	628
eil76	4.8884	578.6667	538
fl417	83.555	13708.59	13538
gil262	35.469	2651.053	2624
gr229	24.483	1871.516	1811
gr431	82.393	2272.486	2192
kroA100	7.3704	23198.51	21282
kroA150	14.0272	29796.38	26524
kroA200	23.1436	33161.28	29368
kroB100	7.4974	23487.86	22141
kroB150	13.9124	28799.06	26130
kroB200	23.0602	33354.33	29437
kroC100	7.4638	21872.4	20749
kroD100	7.4834	23238.09	21294
kroE100	7.5204	24376.85	22068
lin105	7.6322	15247.62	14379
lin318	46.897	48196.07	47816
p654	187.145	42041.29	41065
pcb442	91.956	61484.42	61264
pr107	7.6828	47134.02	44303
pr124	10.0462	62079.37	59030
pr136	11.698	111383.3	96772
pr144	12.3602	60144.43	58537
pr152	13.4956	79603.79	73682
pr299	40.905	55881.29	54231
pr76	4.8174	121464.9	108159
rat195	20.941	2509.549	2323
rat575	144.0207	8209.714	8129
rat99	6.9638	1329.461	1211
rd100	7.4236	8680.192	7910
rd400	80.989	18007.47	17322
st70	4.367	724.5031	675
ts225	25.6224	133592.6	126643
tsp225	25.198	4351.189	3916
u159	14.4444	46391.28	42080
u574	154.176	45063.56	44663
u724	248.335	51069.41	50429

可视化：

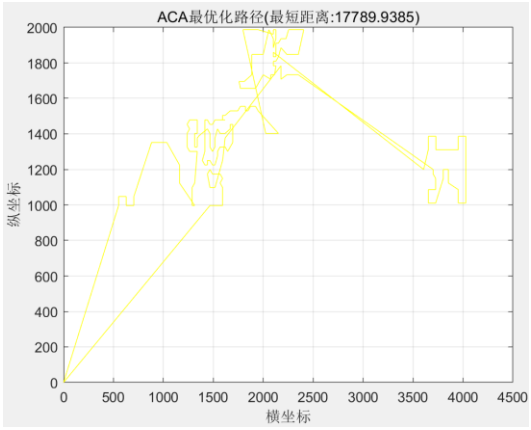

图 3 蚁群算法可视化

4.4.4 遗传算法

参数设置：

种群数：150

进化代数：100

实验结果：

表 4 遗传的结果

类型	时间	距离	最优距离
berlin52	0.278	7717.834	7681.736
bier127	0.599	121975.423	121712.595
ch130	0.610	6281.864	6260.400
ch150	0.725	6861.244	6836.353
d198	0.987	16112.595	16100.570
d493	4.271	37296.713	37295.068
d657	7.436	52878.649	52800.350
eil101	0.552	665.825	661.813
eil76	0.393	563.490	563.135
fl417	2.825	12228.312	12202.826
gil262	1.396	2539.784	2532.869
gr229	0.954	1699.753	1698.803
gr431	3.130	2014.919	2002.002
kroA100	0.537	21698.785	21619.056
kroA150	0.714	27462.417	27441.176
kroA200	0.999	30642.258	30506.498
kroB100	0.519	22702.580	22689.653
kroB150	0.699	27104.922	27033.869
kroB200	0.999	30795.534	30787.773
kroC100	0.519	21186.159	21060.727
kroD100	0.520	21747.732	21586.212
kroE100	0.499	22743.143	22666.428
lin105	0.549	14733.729	14725.309
lin318	1.810	44834.336	44788.022
p654	6.873	35524.045	35515.788
pcb442	3.208	54925.755	54916.335
pr107	0.542	45032.849	44865.408
pr124	0.554	59536.879	59535.384
pr136	0.628	100101.090	99732.433
pr144	0.550	58998.565	58862.087
pr152	0.698	74787.363	74620.769
pr299	1.603	50662.889	50620.261
pr76	0.407	108878.423	108457.399
rat195	0.972	2483.505	2475.592
rat575	5.738	7400.002	7388.769
rat99	0.503	1275.601	1265.625
rd100	0.517	8087.878	8037.936
rd400	2.880	16408.534	16352.709
st70	0.371	686.512	684.539
ts225	1.050	128723.605	128678.181
tsp225	1.144	4074.021	4042.649
u159	0.754	44123.777	43880.360
u574	5.915	39901.270	39756.176
u724	8.968	45672.007	45488.829

可视化：

以 d198.tsp 为数据集,得到的图像如下

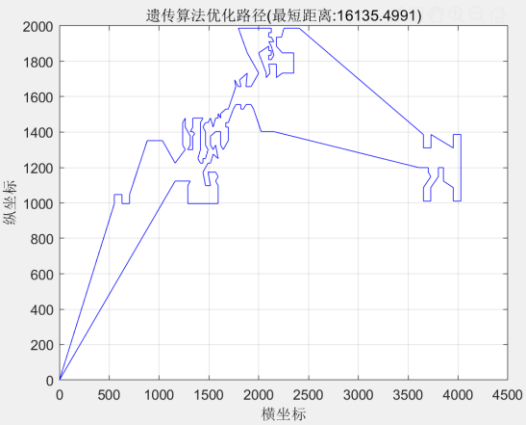

图 4 遗传算法可视化

4.4.5 改进的遗传算法

参数设置：

与 4.4.4 节一致。

实验结果：

表 5 改进后的结果

类型	时间	距离	最优距离
berlin52	0.442629	7651.868	7619.153
bier127	0.865864	121905.46	121599.7
ch130	0.918315	6165.00	6130.948
ch150	1.028261	6801.8	6783.869
d198	1.346936	16073.32	1603854
d493	5.12359	35351.48	35214.78
d657	8.415922	52303.55	52050.76
eil101	0.557048	665.8154	663.4915
eil76	0.591498	567.3286	566.9458
fl417	3.324229	12100.92	12072.94
gil262	1.681804	2538.234	2521.184
gr229	1.5337	1706.442	1698.803
gr431	3.561105	2007.377	2000.808
kroA100	0.772081	21834.91	21780.2
kroA150	1.052616	27115.37	27100.1
kroA200	1.44744	30616.55	30591.51
kroB100	0.741958	22743.47	22727.92

kroB150	1.034863	27087.77	26966.84
kroB200	1.384845	30842.38	30639.55
kroC100	0.732742	21164.81	21123.11
kroD100	0.818393	21826.27	21821.67
kroE100	0.837177	22666.34	22541.19
lin105	0.764409	14645.58	14534.3
lin318	2.322597	42583.46	42363.07
p654	7.959797	33318.58	33100.52
pcb442	3.925028	53653.44	53323.28
pr107	0.770071	45209.96	44957.69
pr124	0.881747	59372.2	59086.81
pr136	0.950734	100434.1	99979.64
pr144	0.948105	58687.49	58653.64
pr152	1.128568	74776.49	74595.39
pr299	2.100826	51200.82	51104.24
pr76	0.595449	109498	109224.3
rat195	1.377997	2486.185	2475.592
rat575	6.483196	7416.58	7380.735
rat99	0.689158	1281.17	1277.569
rd100	0.796201	8095.759	8078.284
rd400	3.45699	16478.58	16470.34
st70	0.559507	685.5621	684.5785
ts225	1.545246	128869	128604.3
tsp225	1.601788	4109.022	4102.589
u159	1.031464	44413.8	42534.13
u574	6.669418	40057.76	39685.27
u724	9.988291	45895.29	45339.26

可视化：

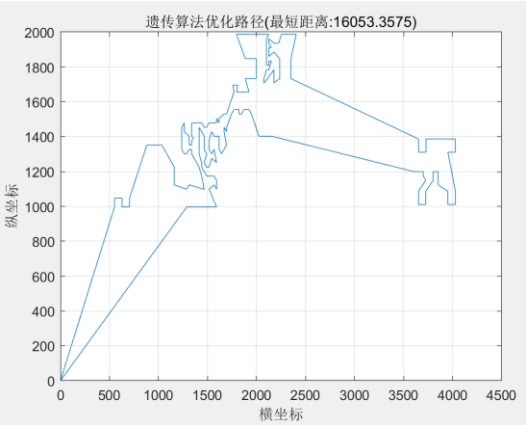

图 5 改进后的可视化

4.3 总体数据及分析

表 6 总体的结果

旅行商问题标准问题库实例	最优路径长度	运行结果					运行时间 (sec)
		MCA [注1]	SA [注2]	ACO [注3]	GA [注4]	GA' [注5]	
berlin52	7542	15968	8401	7812	7682	7619	0.15
bier127	118282	257719	129443	125374	121713	121600	0.13
ch130	6110	13217	6537	6453	6260	6130	0.14
ch150	6528	14720	7271	6881	6836	6784	0.15
d198	15780	33109	16658	17549	16101	16038	0.15
d493	X	75944	52378	40472	37295	35214	0.19
d657	X	108572	97479	59205	52800	52050	0.22
eil101	629	1375	695	711	662	663	0.13
eil76	538	1212	600	579	563	567	0.15
fl417	X	25751	18981	13709	12203	12072	0.18
gil262	X	5123	2770	2651	2533	2521	0.17
gr229	X	3544	1775	1872	1699	1699	0.15
gr431	X	4132	2645	2272	2002	2008	0.17

kroA100	21282	46152	21885	23199	21619	21780	0.13
kroA150	26524	57887	28287	29796	27441	27114	0.15
kroA200	29368	63877	32688	33161	30506	30892	0.16
kroB100	22141	46328	24810	23488	22690	22728	0.13
kroB150	26130	57401	27970	28799	27034	26967	0.16
kroB200	29437	63271	33981	33354	30788	30640	0.15
kroC100	20749	44777	21616	21872	21061	21123	0.14
kroD100	21294	46249	23191	23238	21586	21822	0.13
kroE100	22068	47181	22918	24377	22666	22541	0.14
lin105	14379	30869	15497	15248	14725	14534	0.13
lin318	X	92739	53312	48196	44788	42363	0.17
p654	X	74841	68756	42041	35516	33460	0.24
pcb442	X	112513	75433	61484	54916	54519	0.18
pr107	44303	93233	46924	47134	44865	44958	0.14
pr124	59030	93319	61755	62079	59535	59087	0.14
pr136	96772	209469	105135	111383	99732	99980	0.14
pr144	58537	120995	60770	60144	58862	58654	0.16
pr152	73682	155106	75960	79604	74621	74595	0.16
pr299	X	108908	60547	55881	50620	51104	0.16
pr76	108159	227828	114414	121465	108457	109224	0.13
rat195	2323	5145	2658	2510	2476	2476	0.16
rat575	X	15175	11428	8210	7389	7381	0.20
rat99	1211	2680	1346	1329	1266	1278	0.13
rd100	7910	16681	8671	8680	8038	8078	0.13
rd400	X	33323	21320	18007	16353	16470	0.18
st70	675	1404	729	725	685	685	0.14
ts225	126643	272435	135809	133593	128678	128604	0.16
tsp225	3916	8493	4371	4351	4043	4103	0.16
u159	42080	92205	44991	46391	43880	44277	0.16
u574	X	82162	66300	45064	39756	39685	0.20
u724	X	93815	90129	51069	45489	45339	0.23
近似值[注6]	X	2.5	1.6	1.5	1	0.98	X

- 注 1：改良圈算法
- 注 2：模拟退火算法
- 注 3：蚁群算法
- 注 4：遗传算法
- 注 5：改进的遗传算法
- 注 6：以遗传算法为基准进行近似值的计算，X 为无法计算或未知

经过本次实验我们得出以下结论：

(1) 最终得到的解的优劣排行
遗传算法>蚁群算法>模拟退火算法>改良圈算法

(2) 运行时间上的优劣排行：
改良圈算法>模拟退火算法>遗传算法>蚁群算法

(3) 在小规模的 TSP 问题下，除了改良圈算法，大部分算法都具有比较好的时间性能并能得到较好的解，但蚁群算法相比于其他算法由于搜索的空间较大，每次

蚂蚁都要进行遍历操作，故时间花销会比较大。在大规模的 TSP 问题下，模拟退火算法相比于其他算法都具有极佳的时间性能，但解的优劣程度上会差很多差距。

(4) 改进后的算法在时间上变慢了约 10%，但能得到接近最优解的解，即解的质量提高了 1%-3%。由于改进效果较小且部分数据集中相差不大，不排除是误差的影响。

5 参考文献

[1] Li Qing, Wei Guangcun, Gao Lan, Qiu Guohua, Xiao Xinguang. Improvement of genetic algorithm for solving TSP problem [J]. Software guide, 2020,19 (03): 116-119
(李庆, 魏光村, 高兰, 仇国华, 肖新光. 用于求解TSP问题的遗传算法改进[J]. 软件导刊, 2020, 19(03):116-119.)

[2] Tao Lihua, Ma Zhennan, Shi Pengtao, Wang Ruifeng. Dynamic ant colony genetic algorithm based on TSP problem [J]. Mechanical design and manufacturing, 2019 (12): 147-149 + 154
(江建文, 王纪凯, 陈宗海. 基于GPU的并行遗传算法求解TSP问题[C]. 中国自动化学会系统仿真专业委员会、中国仿真学会仿真技术应用专业委员会、中国科学技术大学. 第二十届中国系统仿真技术及其应用学术年会论文集 (20th CCSSTA 2019). 中国自动化学会系统仿真专业委员会、中国仿真学会仿真技术应用专业委员会、中国科学技术大学: 中国自动化学会系统仿真专业委员会, 2019:560-563.)

[3] Hou Shujing. Comparison of several algorithms for solving TSP problem [J]. Journal of Huanggang Vocational and technical college, 2015,17 (01): 99-102
(侯淑静. 求解TSP问题的几种算法比较[J]. 黄冈职业技术学院学报, 2015, 17(01):99-102.)

[4] Deng Xianxi. Research and improvement of genetic algorithm for solving TSP problem [D]. Northeast University, 2008
(邓先习. 遗传算法求解TSP问题的研究与改进[D]. 东北大学, 2008.)

[5] Wang Jianwen, Dai Guangming, Xie Baiqiao, Zhang Quanyuan. Overview of algorithms for solving TSP problem [J]. Computer engineering and science, 2008 (02): 72-74 + 155
(王剑文, 戴光明, 谢柏桥, 张全元. 求解TSP问题算法综述[J]. 计算机工程与科学, 2008(02):72-74+155.)

[6] Computers and Intractability:A Guide to the Theory of NP-Completeness. Garey M R,Johnson D S. . 1979

[7] GUI Chuanzhi. Application of chaos optimization algorithm in TSP problem [J]. Science and technology innovation guide, 2016,13 (21): 74-75
(桂传志. 混沌优化算法在TSP问题的应

用 [J]. 科 技 创 新 导 报, 2016, 13(21) :74-75.)

[8] Wang Jinfei, Chen Ju, Wei Wei, Li Zhenhua. TSP problem solving based on improved Guo Tao algorithm [J]. Computer engineering and design, 2006 (05): 744-745 + 751
(王劲飞, 陈琎, 魏巍, 李振华. 基于改进郭涛算法的TSP问题求解[J]. 计算机工程与设计, 2006(05) :744-745+751.)

[9] Qi Anzhi. Applied research on TSP problem based on improved simulated annealing algorithm [J]. Information and computer (theoretical Edition), 2020,32 (03): 32-34
(齐安智. 一种基于改进模拟退火算法的TSP问题的应用研究[J]. 信息与电脑(理论版), 2020, 32(03) :32-34.)

[10] https://blog.csdn.net/wys7541/article/details/82387444

I.附录 1——分工

成员	组长	组员	组员	组员	组员
学号	20184484	20184439	20184517	20184544	20184417
姓名	胥卜凡	颜扬升	董修良	王有为	于永勋
班级	计算机 1803	计算机 1803	计算机 1803	计算机 1803	计算机 1803
任务分工	遗传算法核心代码，模拟退火核心代码，测试及分析，各种 TSP 算法的分析,改进的算法设计	蚁群算法核心代码，数据预处理，测试及分析	改良圈算法核心代码，数据预处理，测试及分析	改良圈算法核心代码，数据预处理，测试及分析	蚁群算法核心代码，数据预处理，测试及分析
报告分工	研究过程描述,算法思想设计，绪论，测试数据分析	结果总结，排版	测试数据分析	测试数据分析	测试数据分析

II.附录 2——心得与体会

(1)胥卜凡：

首先感谢炎炎夏日中老师的谆谆教导和组员们的辛勤劳动，这是本次课设成功完成的必要条件。

本次课设可谓是收获满满。首先是我当队长的心得。之前只在进行各种程序设计的时候或者打建模等比赛的时候当过主导的，本次算法设计的队长让我明白了新的协作方式，也就是头脑风暴法。之前一直都是分模块进行人员运作，本次算法课设则把大部分时间用在了小组集体讨论上，每次讨论之后收获都很大，所以我想这也应该是每次团队合作必要的环节吧。

除此以外,还有对于算法设计也有了新的理解和体会。算法设计不是凭空来的，它需要数学基础和对问题的理解能力，而不能仅凭想象就去编写算法。此外，算法设计要面面俱到，要考虑好各种发生的情况而不是简简单单说个设计的思路。最后，优化更是艰难，总是伴随着失败——失败——失败。这可能是我们相关能力水平还是不太高，希望这方面的能力能在未来的学习中得到提升。

最后论文方面的提升也很大,这里的提升既包括了读论文的能力，又包括写论文的能力。先说读论文，我查阅了大量 TSP 相关问题的论文，但要在大量论文中找出我想要的内容，就不能对每篇论文加以细看，否则时间来不急。所以我每次读论文都是先快速浏览摘要，通过每句话的关键字来锁定论文内涵。其次，选定论文后也不能一直细看，而是应该快速浏览找到想要的点，比如优化的具体过程再进行细看。接着说写论文，这可谓是让我煞费苦心，这苦既包括了整篇论文的架构和编写，更包括摘要时写英文的痛苦。不过写完之后成就感满满，感觉对之后要写的各种论文都有了一定的基础，收获颇丰！

最后再一次感谢老师在课设器件对我们的帮助，感谢组员们的努力奋斗。

(2)颜扬升：

首先感谢老师的指导和组员们的付出。本次算法设计可谓是收获满满。

在查阅了 TSP 相关大量资料后,我既巩固了课上所学的动态规划法等内容，又深入了解了目前主流的各种人工智能的算法。此外，通过对蚁群算法的实现，我也意识到我的代码能力还有不足，总是出现莫名奇妙的 BUG,这既有我对 MATLAB 不熟悉的原因，又有对相关算法流程理解不充分的原因，也可能因为这是我第一次通过看论文的流程来实现算法。不过经过本次设计，我相关的能力都得到了很大的提升，相信未来进行相关的操作一定能更熟练！

(3)董修良：

首先感谢老师每次验收时的指导和组员们的付出。

由于我的代码能力较弱,这次我没有参与太多的算法实现内容，但是通过对各种论文的翻阅让我对学习的兴趣有了提升，觉得在科研道路上有很多志同道合的朋友。虽然我只实现了改良圈算法但也遇到了很多莫名其妙的 BUG，但在一一改正之后感

到自己的编程能力有了很大的提升，非常开心。

(4)王有为：

首先感谢老师的指导和组员们的付出。

我学习方面并不理想,所以本次我没有参与太多内容的实现，但是我也收获了许多东西。一方面我对 TSP 相关问题有了新的理解，既包括对 NP 问题算法设计、相关复杂度的理解，又包括对人工智能算法的理解；另一方面我对论文查询也有了理解，知道了如何通过学校的相关连接来进行论文查询，从而扩大想要学习的知识面的前沿。

(5)于永勋：

首先感谢老师的指导和成员们的互帮互助。

本次设计中我负责的是蚁群算法的实现。在设计过程中遇到了很多困难，首先就是读取数据让我卡壳了，在和组员们讨论之后终于知道了比较好的读取数据的方法。之后在实现的过程中，我还遇到蚂蚁路径混乱交叉的问题，最终通过逐行调试成功解决了问题，原来是一个变量放错了位置，这让我明白了对于程序来说，一个小小的失误就会引发巨大的连锁反应，所以编写时一定要仔细认真。此外，我还对 TSP 算法有了更深的理解，包括 NP-Hard 问题等等。

III.附录 3——失败的改进之一

思想：针对选择算子，采用动态的适应度函数计算。

即通过下述算子对适应度进行运算：

$$F=\frac{f+f_{\min}}{f_{\max}+f_{\min}+\alpha},\quad (\alpha>0)\ ,$$

(0.6)

其中 f 是某一个体的适应度值，$f_{\min}$ 和 $f_{\max}$ 分别代表本代种群中的最小和最大适应度值，α 是用来防止出现分母为 0 的情况。 基本思想：当最大适应度和最小适应度差别小时，适应度值的范围扩大，提高选择能力，防止过度收敛；当差别较大时，降低选择能力，防止过度发散。

优化过程：将算法中的选择操作前增加将路径进行优化

结果：没有优化成功反而负优化。

分析：编写的代码前期收敛过快导致提前早熟，还想使用动态的思想可能需要重新架构。