多种算法求解 TSP 问题的性能分析

组长：20184411 眭永熙 组员：20184545 岳崇建 组员：20184501 王晓菲 组员：20184594 鲁风

摘 要 针对 TSP 问题，实现了最近邻算法、2-opt 算法、最短链接算法、遗传算法、禁忌搜索算法、蚁群算法和模拟退火算法，对 benchmark 中的 30 种 TSP 实例以及 gr229(地理距离)进行了实验与分析。使用 GUI 与可视化辅助算法改进，改进后的蚁群算法在求解精确度和稳定性上都有出色的表现；而模拟退火算法有着速度快且精确度相对较高的优点。

关键词 TSP 问题;最近邻;2-opt;最短链接;遗传算法;禁忌搜索;蚁群算法;模拟退火;可视化

Performance analysis of several algorithms for TSP

Abstract To solve the TSP problem, implemented the nearest neighbor algorithm, 2-OPT algorithm, shortest link algorithm, genetic algorithm, tabu search algorithm, ant colony algorithm and simulated annealing algorithm. Using GUI and visualization to improve algorithm, the improved ANT colony algorithm has excellent performance in solving accuracy and stability. The simulated annealing algorithm has the advantages of high speed and relatively high accuracy.

Key words TSP ; Nearest neighbor; 2 - opt; Shortest link; GA; Tabu search; MMAS; SA; visualization

1 引言

旅行商问题(Traveling Salesman Problem，简称 TSP) 的描述很简单，但是很难解决。TSP 问题可以如下说明。考虑 n 个城市之间的距离，从一个城市到各城市的旅行商只能访问一次，最后回到出发城市，求最短路线。

在现实生活中，有许多问题可以归因于旅行商的问题，如邮政路线问题、流水线上的螺母问题和产品的生产安排问题。Tsp 在 PCB 钻孔调度、基因测序和机器人控制等方面有着重要的应用。因此，求解 TSP 问题具有重要的理论和现实意义。

数十年前发现的一些指数算法可以准确地解决 TSP，但随着问题的规模扩大，这些算法将完全失败（组合爆炸）。虽然近似算法和启发式算法不能正确解决问题，但是因为可以在允许范围内控制误差，所以可以快速得到解答。

本文将通过几种贪婪算法、局部搜索算法和近似算法，求解 TSP 并分析它们的性能。

2 相关概念与工作

自 1759 年欧拉研究骑士周游问题提至今，TSP 问题已有各种各样的求解算法。

动态规划法 DM，由美国数学家 Bellman RE 等人在 20 世纪 50 年代初在研究多阶段决策过程的优化问题时提出的[1]，它的时间复杂度为 $O(N^2 \cdot 2^N)$， 空间复杂度为 $O(N \cdot 2^N)$。

分枝限界法 LC，由达金 RJ 和兰德-多伊格在 20 世纪 60 年代初提出[2]，它的时间复杂度为$O(N^2 \cdot 2^N)$。

遗传算法，由美国的 J·Holland 教授 1975 年首先提出[3]。遗传算法的复杂度主要由选择算子、交叉算子以及变异算子的时间复杂度决定，根据不同实现方法具有不同的复杂度，一般情况下时间复杂度为$O(K \cdot N^2)$。

郭涛算法，由武汉大学的郭涛博士于 1999 年提出的一种对遗传算法的改进算法[4]。

人工神经网络，由 Hopfield 和 Tank 在 1985 年提出[5]， 通过对神经网络引入适当能量函数，使之与 TSP 的目标函数相一致来确定神经元之间的联结权。

模拟退火，由 Patrick K 于 1982 年提出[6]，模拟物理中固体物质的退火过程来求解 TSP 问题。

蚁群算法，由意大利学者 Dorigo M 等在 1996 年首先提出[7]，通过模拟蚂蚁觅食过程求解 TSP 问题。

禁忌搜索算法，最早由 Fred Glover 教授提出[8]，是对人类智力过程的一种模拟。

Lin-Kernighan 启发式算法，由 Lin-Kernighan 于 1973 年提出[9]的一种试探优化法。

粒子群优化 PSO 算法，由 J. Kennedy 和 R.C. Eberhart 在 1997 年提出[10]，用粒子群的思想求解 TSP 问题。

3 算法设计

3.1 贪心算法

贪心算法是指在对问题求解时，总是做出在当前看来是最好的选择，不从整体最优上加以考虑，他所做出的是在某种意义上的局部最优解，不能保证最后结果是最优的，但能保证是比较优秀的，贪心算法的效率高。

贪心选择性值是指，所求问题的整体最优解可以通过一系列局部最优的选择，即贪心选择来达到。贪心算法所做的贪心选择可以依赖以往所做过的选择，但决不依赖将来所做的选择，也不依赖子问题解。

3.1.1 最近邻点

3.1.1.1 原理

最近邻点策略：从某一城市出发，下一次访问最近邻的，未访问过的城市，然后从该城市出发，选择一个离它最近的，未访问过的城市，重复上述过程直到所有城市都被访问，最

后回到出发城市。

3.1.1.2 例子

如图 1-1-1

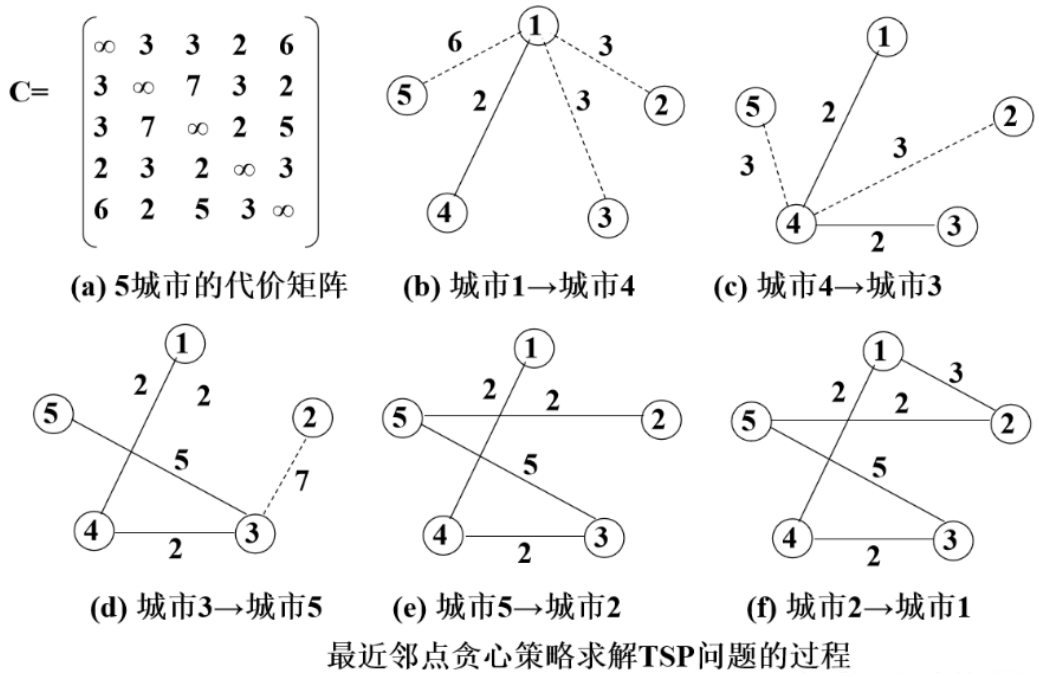


图 1-1-1 最近邻点算法原理简单示例

3.1.1.3 伪代码

设置出发城市；
初始化访问数组为 0；
循环判断是否已经结束{
　　循环找距离当前城市最近的下一个城市{
　判断该城市是否被访问过
　　如果距离小于当前最小距离，那么记录该城市，将这个距离设置为新的最小距离}
访问该城市；
总距离+=距离；
记录访问路径；
更新当前城市；}
回到出发城市；
总距离+=距离；
记录访问路径；
输出访问路径；
输出总距离；

3.2 2-OPT 算法

一般情况下，找最优路径采取贪心策略，都可以收到一个比较好的成效，但是如果想要得到一个更优的解 ，这时候需要对贪心策略进行优化。但其实用一个算法找到最优解是一个非常困难的问题，一般会用一个概率上的优化，就是如果迭代次数足够多，找不到一个更好的解，就认为这个解是最优的。

3.2.1 原理

首先任选一个可行解，然后用 2-OPT 算法进行问题求解，随机选取两点看 k,m,将 k 之前路径不变添加到新路径中，将 k 到 m 之间的路径翻转添加到新路径，将 k 之后的路径不变添加到新路径中。从而获得一个新的可行解，如果新的可行解比原来的解更加优化，就保留新的可行解。重复上述做法，直到交换 N 次仍然无法更新可行解。

3.2.2 伪代码

```
设置迭代次数
循环判断是否结束迭代{
随机数 k;
随机数 m;
如果(k>m){交换 k,m}
如果(m=n){计算 k-1 到 k，m 到起始点路径与 k-1 到 m，k 到起始点路径的差值}
else{计算 k-1 到 k，m 到 m+1 路径与 k-1 到 m，k 到 m+1 点路径的差值}
如果差值大于 0{将 m 和 k 之间的路径翻转;
                计数清 0;
                总距离-=差值;}
                计数++;}
输出路径；
输出总距离；
```

3.3 最短链接

3.3.1 原理

最短链接策略：最短链接是基于贪心算法，每次在整个图的范围内选择最短边加入到解集和中。为了保证加入解集和的边最终形成一个哈密尔顿回路，因此每次从剩余边集 E’中选择

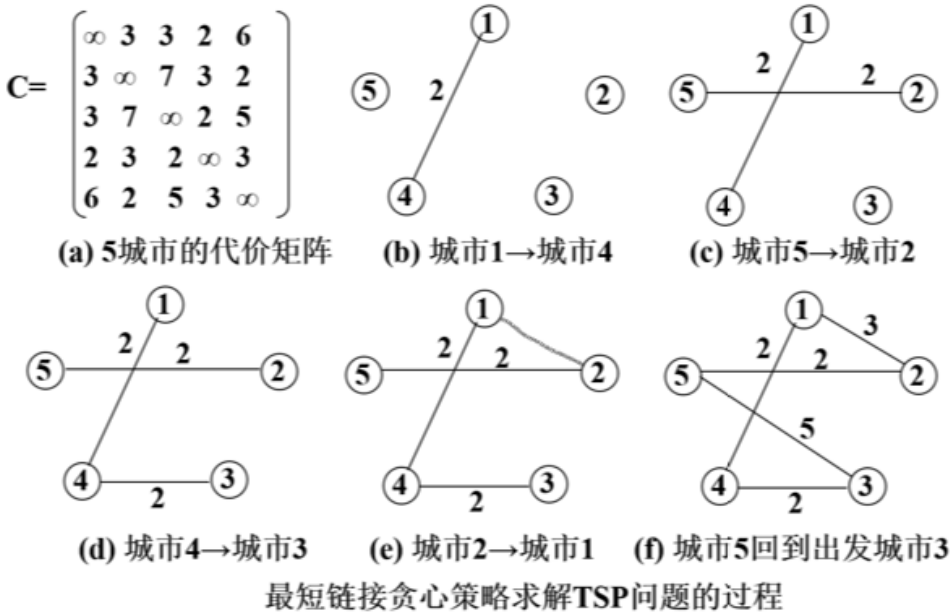
一条边（u,v）加入解集和 S 中，应满足如下条件：

（1）边（u,v）是边集 E’中代价最小的边；

（2）边（u,v）加入解集和 S 后，S 中不产生回路；

（3）边（u,v）加入解集和中，S 中不产生分枝

3.3.2 例子



最短链接贪心策略求解TSP问题的过程

3.3.3 算法及伪代码

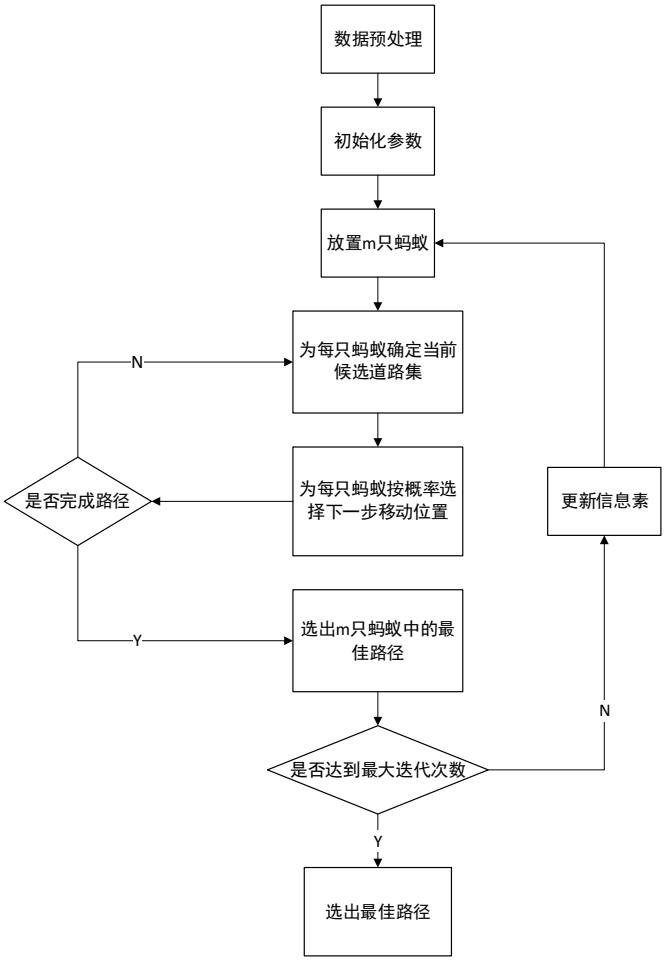
设图 G 中有 n 个顶点，二维数组 dis[n][n]用来存储边上的代价，E’是候选集合，存储所有未选取的边，集合 P 存储经过的边，算法如下：

```
P={}
E’=E  //初始化
While(P.length <= n-1)  //集合 P 中边数少于 n-1 条时，循环加入新边
{
  Sort(E’)   //将 E’中边按代价从小到大排序，选取代价最小的边（u,v）
  E’=E’-{（u,v）};
  If (u 和 v 在 P 中不连通并且不产生分枝）
    P=P+{（u,v）};
}
```

3.4 MMAS 算法

生物学家经过大量的观察研究发现，蚂蚁个体之间通过一种化学激素(信息素)进行信息交流与协作的。蚂蚁在运动过程中，能够在它所经过的路径上留下该化学物质，并能够感知这种物质。一条路径上的信息素量越大，对蚂蚁的吸引力就越大，选择该路径的概率就越高，从而使得该路径的信息素浓度得到进一步加强，表现出一种信息正反馈现象。蚂蚁群体就是通过这种间接的通讯机制达到了在觅食过程中沿着最短路径前进的目的。

3.4.1　流程图



3.4.2　蚁群算法基本公式

3.4.2.1 城市选择概率公式

设有 m 只蚂蚁随机放在具有 n 个节点的全连通图上，$\tau_{i,j}(t)$表示 t 时刻在i,j连线上的信息素量，$\eta_{i,j}(t)$是与问题相关的启发式信息，在 TSP 问题中，一般取为$\frac{1}{d_{i,j}}$ ($d_{i,j}$ *为点*i,j*之间的距离*)。初始时刻，各条路径上信息量相等，设$\tau_{i,j}(0)=C$ (C*为常数*)。蚂蚁 k $(k=1,2,\dots,m)$在运动过程中根据各条路径上的信息量和问题的启发式信息决定转移方向。

式$P_{i,j}^{k}(t)$表示在t时刻蚂蚁k由位置i转移到位置j的概率，

$$P_{i,j}^{k}(t)=\begin{cases}\dfrac{\left[\tau_{i,j}(t)\right]^{\alpha}\cdot\left[\eta_{i,j}(t)\right]^{\beta}}{\sum_{s\notin tabu_k}\left[\tau_{i,s}(t)\right]^{\alpha}\cdot\left[\eta_{i,j}(t)\right]^{\beta}}, & j\notin tabu_k\\ 0, & j\in tabu_k\end{cases}$$

式中，$tabu_k$记录着蚂蚁已经遍历的城市，α, β为调节信息素强度τ和启发式信息η相对重要性的参数。

3.4.2.2 信息素更新公式

经过 n 个时刻，蚂蚁完成一次周游，各路径上信息量要根据下式作调整:

$$\tau_{i,j}(t+n) = (1-\rho)\cdot\tau_{i,j}(t) + \Delta\tau_{i,j}(t+n)$$

$$\tau_{i,j}(t+n) = \Sigma_{k=1}^{m}\Delta\tau_{i,j}^{k}(t+n)$$

式中，$\rho \subset (0,1)$表示信息素的挥发系数，$\Delta\tau_{i,j}^{k}$表示第 k 只蚂蚁在本次循环中留在路径(i,j)上的信息量；$\Delta\tau_{i,j}$表示本次循环中所有蚂蚁留在路径(i,j)上的信息量。

$$\Delta\tau_{i,j} = \begin{cases} \dfrac{1}{L_{min}}, & (i,j) \in BestPath \\ 0, & Others \end{cases}$$

将信息素的值限制在[τ min , τ max]之间，对由于信息素挥发和更新导致的信息素浓度偏高、偏低的值都强行限制在[τ min , τ max]之间，避免算法过早收敛至局部最优解。

3.4.3 伪代码

蚁群优化算法
输入：城市坐标数组
输出：最优路径及其长度

```
PROCEDURE AS
    FOR EACH edge
        初始化信息素 Tau、启发式值 Eta
    END FOR
    WHILE NOT STOP
        FOR EACH ant k
            随机选择一个城市
            FOR i=1 TO n
                根据概率公式，采取轮盘赌方式选择下一步到达的城市
                //该城市不在禁忌集 Tabu(k)中
                将该城市加入到禁忌集 Tabu(k)中
            END FOR
        END FOR
        FOR EACH ant k
            根据 Tabu(k)计算蚂蚁 k 所走的路径长度
        END FOR
        记录最短路径
        FOR EACH edge
            根据公式更新信息素
        END FOR
        清空 Tabu
    END WHILE
    PRINT result
END PROCEDURE
```

3.5 模拟退火算法

模拟退火算法的思想最早由 Metropolis 等提出的。其出发点是基于物理中固体物质的退火过程与一般的组合优化问题之间的相似性。

其中 Metropolis 准则是 SA 算法收敛于全局最优解的关键所在,Metropolis 准则以一定的概率接受恶化解，这样就使算法跳离局部最优的陷阱。

3.5.1 基本公式

3.5.1.1 Metropolis 准则

假设前一个状态为$x(n)$,系统根据某一指标（目标函数），状态变为$x(n+1)$,相应的，系统的能量由$E(n)$变为$E(n+1)$,定义系统由$x(n)$变为$x(n+1)$的接受概率P为：

$$P=\begin{cases}1 & ,E(n+1)<E(n)\\ e^{-\frac{E(n+1)-E(n)}{T}} & ,E(n+1)\geq E(n)\end{cases}$$

3.5.1.2 退火公式

假设当前迭代时温度为$T(N)$，下一次迭代温度降为$T(N+1)$，则按以下公式退火：

$$T(N+1)=\lambda T(N),\qquad N=1,2,3,\ldots\ldots$$

其中λ是小于 1 的正数，一般取值为 0.8 到 0.99 之间。

3.5.2 伪代码

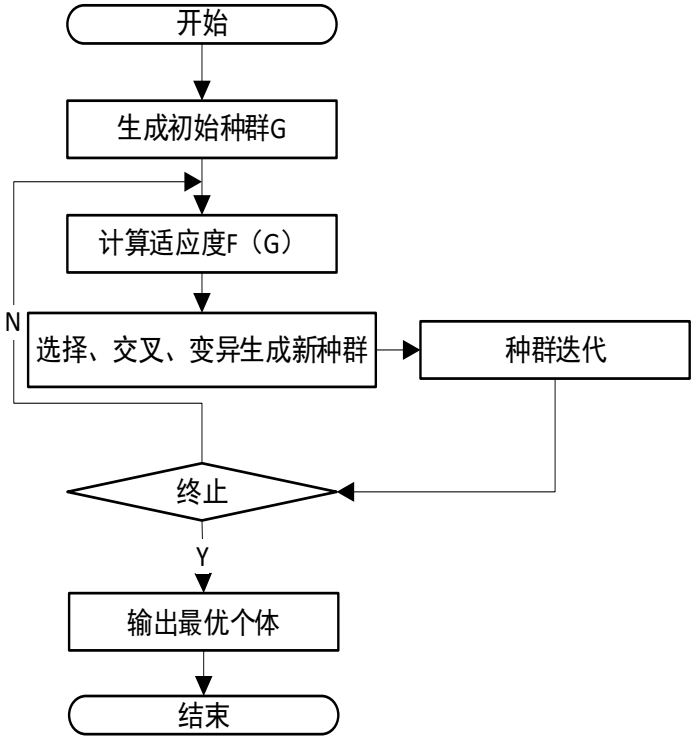
模拟退火算法
输入：城市坐标数组
输出：最优路径及其长度
设置初始温度 T，设置初始解 S

```
PROCEDURE SA
    WHILE T < 终止温度
        WHILE 外层循环未达到迭代上限
            产生新解 S'
            ΔT=C(S')- C(S)，其中 C(S)为目标函数
            IF ΔT < 0
                S = S'
            ELSE
                以概率 exp(-ΔT/T)接受 S'作为新的当前解
            END IF
            如果连续若干个新解都没有被接受时结束算法，输出当前解为最优解
        END WHILE
        T = λT
```

```
    END WHILE
END PROCEDURE
```

3.6 遗传算法

遗传算法是一类借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化搜索方法。它是由美国的 J.Holland 教授 1975 年首先提出，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；具有内在的隐并行性和更好的全局寻优能力；采用概率化的寻优方法，能自动获取和指导优化的搜索空间，自适应地调整搜索方向，不需要确定的规则。



3.6.1 优化策略

1：精英保留策略。自然界中最优秀的个体具有更强的生存能力，所以保留每一代中的最优个体直接进入下一代

2：交叉方式变换。自然界中基因的交叉具有多种方式，所以当连续多代种群的最优个体没有出现变化时，更改种群的交叉方式，即在 PMX 和 OX 交叉方式之间进行互换。

3：变异概率提升。若种群长期不变化，则种群的变异概率开始逐渐升高，直至最优个体出现更新，变异概率恢复。

4：即时停止策略。当连续多代种群的最优个体没有出现变化时停止种群迭代，此时种群极大概率收敛。

5：蒙特卡洛法生成初始种群。随机生成若干个体并根据适应度大小排序，选择其中最优的前几个作为初始种群。

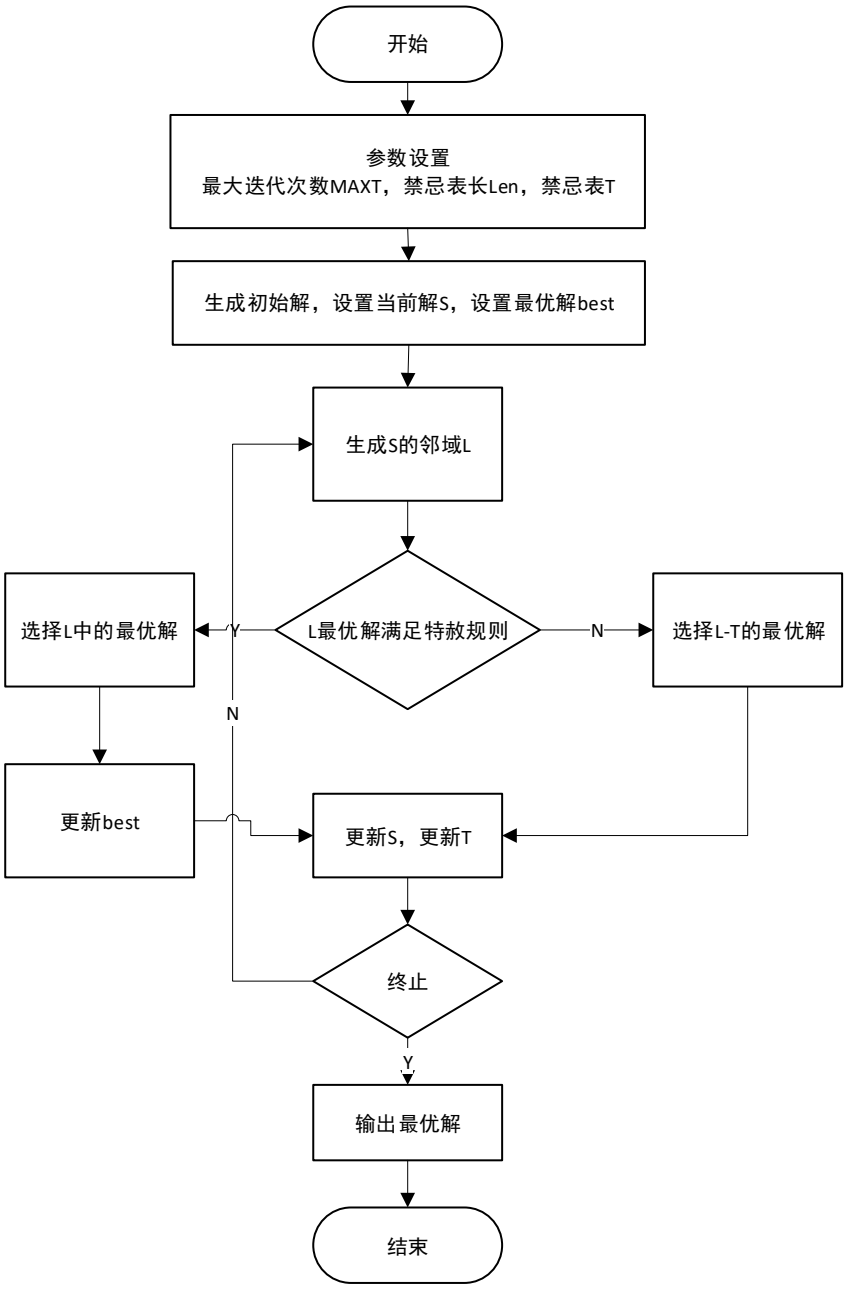
6：贪心优化。在初始种群中加入贪心策略生成的个体，加快种群收敛的速度，但有可能陷入局部最优。

3.7 禁忌搜索

禁忌搜索（Tabu Search 或 Taboo Search，简称 TS）的思想最早由 Fred Glover(美国工程院院士，科罗拉多大学教授)提出，它是对局部领域搜索的一种扩展，是一种全局逐步寻优算法，是对人类智力过程的一种模拟。

TS 算法通过引入一个灵活的存储结构和相应的禁忌准则来避免迂回搜索，并通过藐视准则来赦免一些被禁忌的优良状态，进而保证多样化的有效探索以最终实现全局优化。相对于模拟退火和遗传算法，TS 是又一种搜索特点不同的 meta-heuristic 算法。

3.7.2 流程图



3.7.3 优化点

1、邻域选择。路径上两点交换产生新路径，通过随机选择交换的两点，减小邻域的大小加快迭代速度。
2、禁忌对象。禁忌对象为路径上互换的两个点。
3、特赦规则。当邻域中的最优解比历史上的最优解还优秀时，即使其处于禁忌表中仍可以使用这个解进行迭代。
4、初始路径。使用蒙特卡洛法生成初始路径，或者以其他算法生成的路径作为初始路径。

4 实验结果与分析

4.1 实验环境

CPU: Inter Core i7-10700 2.9GHz eight-core
RAM: 16 GByte

4.2 实验结果

实验测试了 benchmark 中的所有实例，同时测试了 benchmark 中未包含的 gr229 实例。该实例采用地理距离计算，实验的测试框架可以根据不同的实例选择不同的距离计算方式。

以下图表中使用 Nearest 指代最近邻算法，Chain 指代最短链接算法，SA 指代模拟退火算法，GA 指代遗传算法，Tabu 指代禁忌搜索算法，AS 指代最大最小蚁群算法。

4.2.1 实验一

		2-opt		Nearest		Chain	
TSPLIB	Optimal	Length	Time	Length	Time	Length	Time
d198.tsp	15780	18670	0.029	18240	0.027	18933	0.008
bier127.tsp	118282	129173	0.017	135737	0.015	135041	0.005
berlin52.tsp	7542	8235	0.008	8980	0.006	9951	0.003
ch130.tsp	6110	7417	0.018	7579	0.016	7200	0.005
ch150.tsp	6528	8187	0.019	8191	0.018	7809	0.006
eil101.tsp	629	728	0.013	803	0.013	732	0.004
eil76.tsp	538	621	0.01	642	0.008	603	0.004
kroA100.tsp	21282	24763	0.013	27807	0.012	24287	0.004
kroA150.tsp	26524	30918	0.02	33633	0.017	31892	0.005
kroA200.tsp	29368	41618	0.026	35859	0.024	34601	0.008
kroB100.tsp	22141	23886	0.013	29158	0.012	25813	0.004
kroB150.tsp	26130	31444	0.021	34499	0.018	31427	0.005
kroB200.tsp	29437	36729	0.027	36980	0.024	35975	0.007

kroC100.tsp	20749	23996	0.013	26227	0.012	23295	0.004
kroD100.tsp	21294	24147	0.013	26947	0.012	24448	0.004
kroE100.tsp	22068	24760	0.013	27460	0.012	24846	0.004
lin105.tsp	14379	16118	0.015	20356	0.012	16689	0.005
pr107.tsp	44303	49657	0.015	46680	0.015	47933	0.004
pr124.tsp	59030	75906	0.016	69297	0.014	64998	0.005
pr136.tsp	96772	109510	0.018	120769	0.016	117513	0.005
pr144.tsp	58537	76901	0.019	61652	0.015	65844	0.005
pr152.tsp	73682	86098	0.02	85699	0.018	84879	0.006
pr76.tsp	108159	130613	0.01	153462	0.009	140349	0.003
rat195.tsp	2323	2864	0.025	2752	0.023	2787	0.007
rat99.tsp	1211	1416	0.013	1554	0.011	1513	0.004
rd100.tsp	7910	8953	0.013	9938	0.012	9249	0.004
st70.tsp	675	727	0.009	830	0.008	789	0.004
ts225.tsp	126643	197178	0.029	152493	0.027	144228	0.008
tsp225.tsp	3916	5256	0.028	5030	0.027	4780	0.008
u159.tsp	42080	43135	0.019	54675	0.019	48654	0.006
gr229.tsp	134602	174951	0.041	169713	0.038	151973	0.016
		Mean opt=1.194		Mean opt=1.24		Mean opt=1.173	

表 1 实验 1 结果

实验中，测试了 2-opt 算法、最近邻算法、最短链接算法的性能。这三种算法的特点是速度快，但是解的准确度有待提高。因此，考虑将这三种算法作为近似算法的混合优化策略。

4.2.2 实验二

		SA		GA		Tabu	
TSPLIB	Optimal	Length	Time	Length	Time	Length	Time
d198.tsp	15780	16230	0.177	18216	3.158	16335	20.777
bier127.tsp	118282	120620	0.127	130253	3.239	121568	8.284
berlin52.tsp	7542	7944	0.075	8310	0.142	7944	1.223
ch130.tsp	6110	6625	0.136	7039	3.454	6623	8.654
ch150.tsp	6528	6723	0.148	7617	2.338	6686	11.629
eil101.tsp	629	639	0.11	700	2.24	643	4.8
eil76.tsp	538	572	0.093	603	0.456	574	2.664
kroA100.tsp	21282	22652	0.108	25497	1.477	22715	4.968
kroA150.tsp	26524	28850	0.146	30572	3.448	28681	11.597
kroA200.tsp	29368	30401	0.181	34503	4.443	31096	21.342
kroB100.tsp	22141	22477	0.108	25562	2.511	22882	4.97
kroB150.tsp	26130	29120	0.145	30373	7.214	27199	11.68
kroB200.tsp	29437	32597	0.179	35403	8.066	31842	21.279
kroC100.tsp	20749	22673	0.108	23801	1.561	21647	4.968
kroD100.tsp	21294	22253	0.108	24321	1.889	21889	4.96
kroE100.tsp	22068	24429	0.109	24941	1.579	23188	4.954
lin105.tsp	14379	16365	0.113	16234	2.013	15327	5.452

pr107.tsp	44303	44577	0.113	44988	2.846	44600	5.443
pr124.tsp	59030	62274	0.125	65989	2.224	61823	7.666
pr136.tsp	96772	106606	0.132	116780	2.52	106540	9.322
pr144.tsp	58537	61244	0.139	61615	0.967	61244	10.427
pr152.tsp	73682	75039	0.144	82755	2.451	75022	11.918
pr76.tsp	108159	113437	0.091	126818	1.154	115994	2.771
rat195.tsp	2323	2484	0.174	2661	4.851	2417	19.329
rat99.tsp	1211	1324	0.108	1391	1.265	1328	4.712
rd100.tsp	7910	8376	0.109	9105	1.819	8417	4.959
st70.tsp	675	718	0.089	782	0.401	726	2.275
ts225.tsp	126643	136515	0.195	149239	6.529	129796	26.532
tsp225.tsp	3916	4208	0.197	4740	6.265	4241	26.084
u159.tsp	42080	46902	0.149	50545	4.053	46630	12.957
gr229.tsp	134602	150401	0.206	164881	23.847	148562	98.428
		Mean opt=1.066		Mean opt=1.148		Mean opt=1.056	

表 2 实验 2 结果

		AS	
TSPLIB	Optimal	Length	Time
d198.tsp	15780	15848	82.403
bier127.tsp	118282	119110	24.06
berlin52.tsp	7542	7542	2.125
ch130.tsp	6110	6180	28.306
ch150.tsp	6528	6559	39.161
eil101.tsp	629	630	12.86
eil76.tsp	538	544	5.92
kroA100.tsp	21282	21389	12.525
kroA150.tsp	26524	27075	39.097
kroA200.tsp	29368	29700	88.648
kroB100.tsp	22141	22307	12.529
kroB150.tsp	26130	26328	39.218
kroB200.tsp	29437	29812	88.959
kroC100.tsp	20749	21018	12.587
kroD100.tsp	21294	21309	12.556
kroE100.tsp	22068	22121	12.624
lin105.tsp	14379	14401	14.015
pr107.tsp	44303	44570	14.605
pr124.tsp	59030	59552	21.91
pr136.tsp	96772	99241	28.782
pr144.tsp	58537	58669	33.437
pr152.tsp	73682	74249	39.157
pr76.tsp	108159	110030	5.798
rat195.tsp	2323	2332	78.164
rat99.tsp	1211	1212	11.925

rd100.tsp	7910	7936	12.574
st70.tsp	675	679	4.744
ts225.tsp	126643	128284	117.865
tsp225.tsp	3916	3952	117.934
u159.tsp	42080	42291	43.718
gr229.tsp	134602	135450	123.584
		Mean opt=1.007	

表 3 实验 2 结果

实验测试了模拟退火算法，禁忌搜索算法，蚁群算法和遗传算法这四种算法。其中，对模拟退火算法、禁忌搜索算法和蚁群算法都进行了改进。

值得注意的是，改进后的蚁群算法表现出色，在 berlin52 实例中能够稳定的找到最优解，但是相应的时间代价比其他三种算法都要大。而模拟退火算法有着运行时间短，求解比较精确的优点。

实验中对上述算法进行了调参。

4.2.2.1 模拟退火算法调参

模拟退火算法的参数主要有初始温度、退火速率、结束温度、内循环次数，其中前三个参数控制着模拟退火算法的外循环迭代次数。增加内循环迭代次数，发现算法并没有得到明显的改进，当迭代次数大于 200 后，还出现了负优化的现象。原因是内循环很快达到了局部最优的状态，因此内循环迭代次数控制为 100 次。增加外循环迭代次数，算法在 700 次左右收敛。

4.2.2.2 最大最小蚁群算法调参

蚁群算法参数主要有蚁群数量 m，信息素重要度 α，启发式重要度 β，信息素挥发率 ρ。算法设置蚁群数量 m 为城市数量的 1.5 倍。由于 α 和 β 是相对的参数，不失一般性，设置 α 为 1。调整参数 β，发现 β 参数过小和过大都会使得算法效果下降，推测是陷入了局部最优，最后调整 β 为 7。信息素挥发率 ρ 设置的过大会使算法收敛速度变慢，在迭代次数内不能达到收敛状态，而过小会让算法过早收敛，最后在实验测试下调整 ρ 为 0.15。

4.2.3 实验数据图表分析

将算法的解除以实例的已知最优解，得到相对精确度，这个值越接近 1，表明算法的解越接近最优解。

图 4-2-1 展示了各个算法在 KroD100 实例中的表现，可以看到蚁群算法的解非常接近 1，但是相对的时间代价是所有算法中最高的。而模拟退火算法运行时间短，同时求解也较为精确。

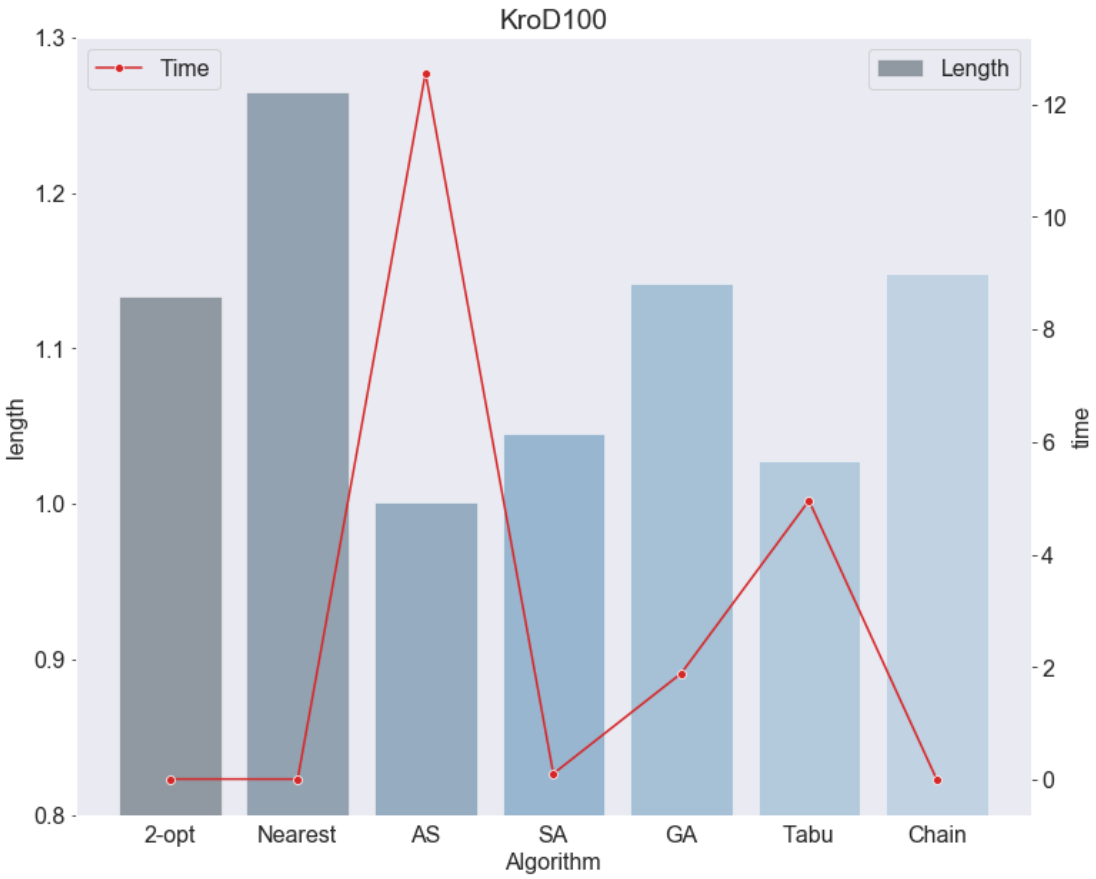


图 4-2-1 算法在 KroD100 实例中的性能

图 4-2-2 展示了各个算法的平均相对精确度，不同算法之间形成了明显的梯度

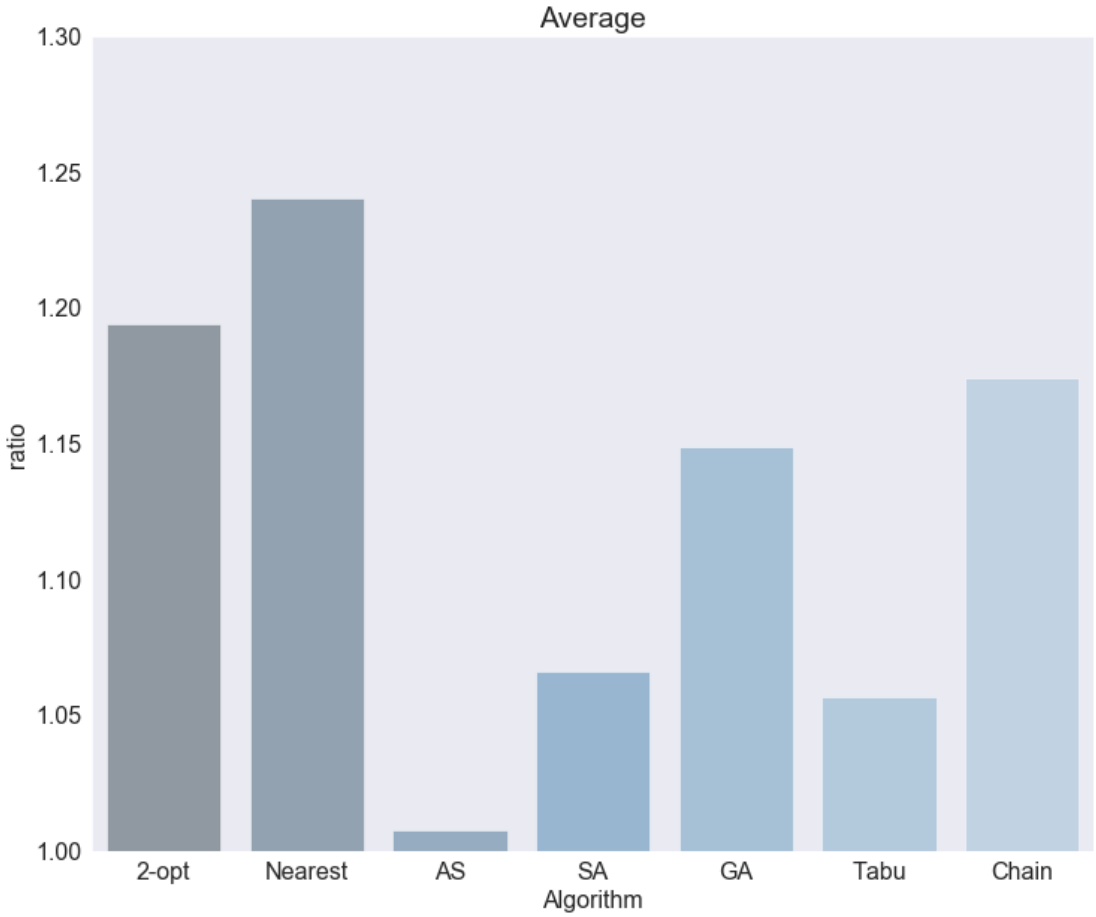


图 4-2-2 算法的平均近似性能

图 4-2-3 展示了各个算法在不同实例中的精确度变化折线，蚁群算法不仅精确度最好，同时曲线也几乎是一条平稳的直线。图 4-2-4 以箱型图的形式展示了这些算法的数据分布，进一步显示了不同算法的稳定性。

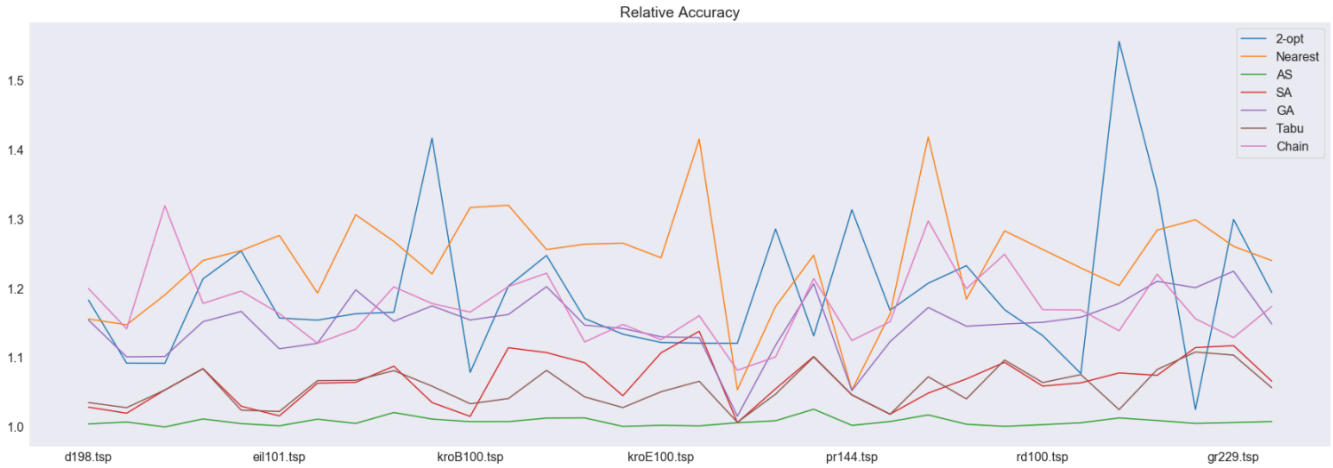


图 4-2-3 算法在不同实例中的近似性能

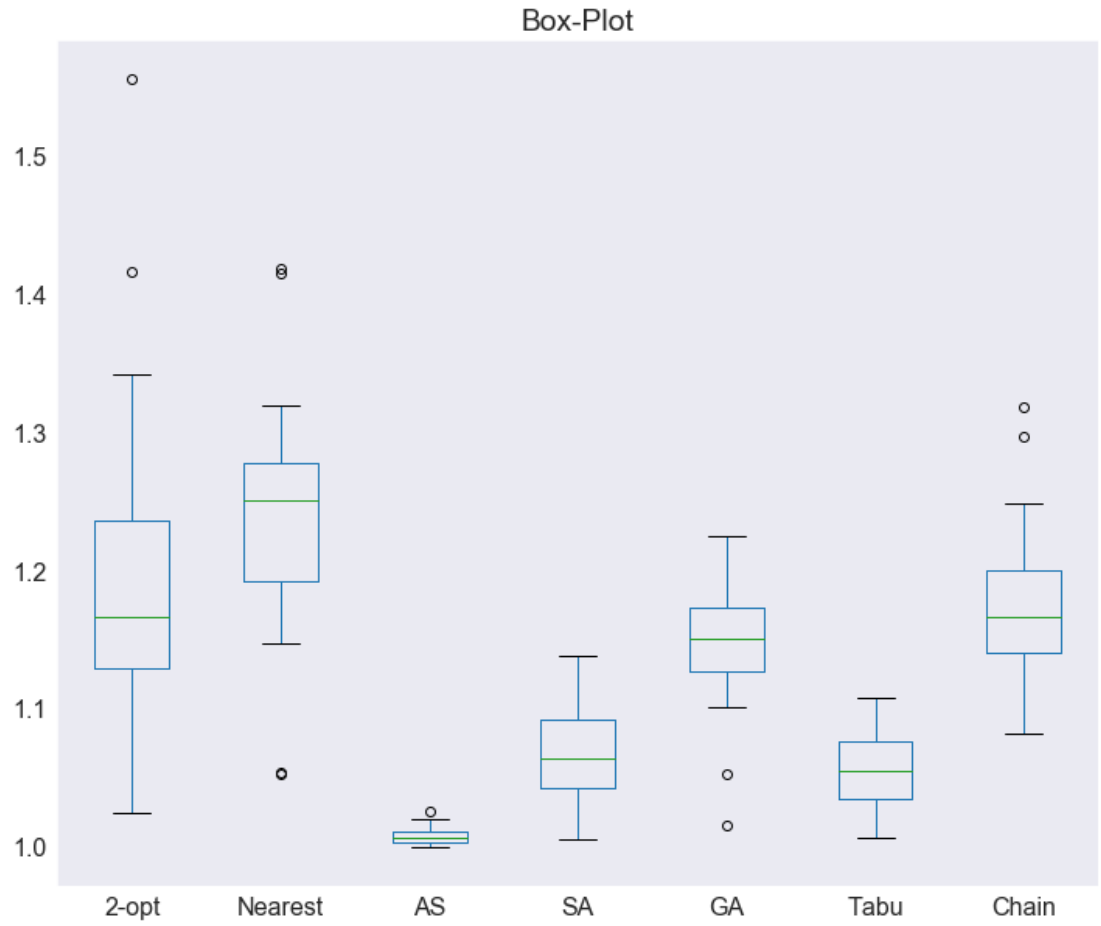


图 4-2-4 算法在所有实例中的近似性能分布

4.3 实验结论

如果对于精确度要求较高而对于时间要求不高，可以选择最大最小蚁群算法；如果对于时间要求较高，可以选择模拟退火算法。

5 算法改进优化

5.1 蚁群算法优化

改进思路：将 2-opt 算法和蚁群算法混合
假设蚂蚁数量为 m，在所有蚂蚁完成一次周游后，选出路径长度最短的 m/2 只蚂蚁，对这些蚂蚁的路径使用 2-opt 算法优化，然后再更新全局的信息素。
蚁群算法中，不是所有蚂蚁都对路径的优化有贡献，因此对于没有贡献的蚂蚁使用 2-opt 优化路径将造成时间上的浪费。
图 5-1-1 展示了优化前后蚁群算法在各个实例以及平均情况下的表现，可以看出改进前后时间差基本不超出 1s，同时优化的相对幅度也比较大。

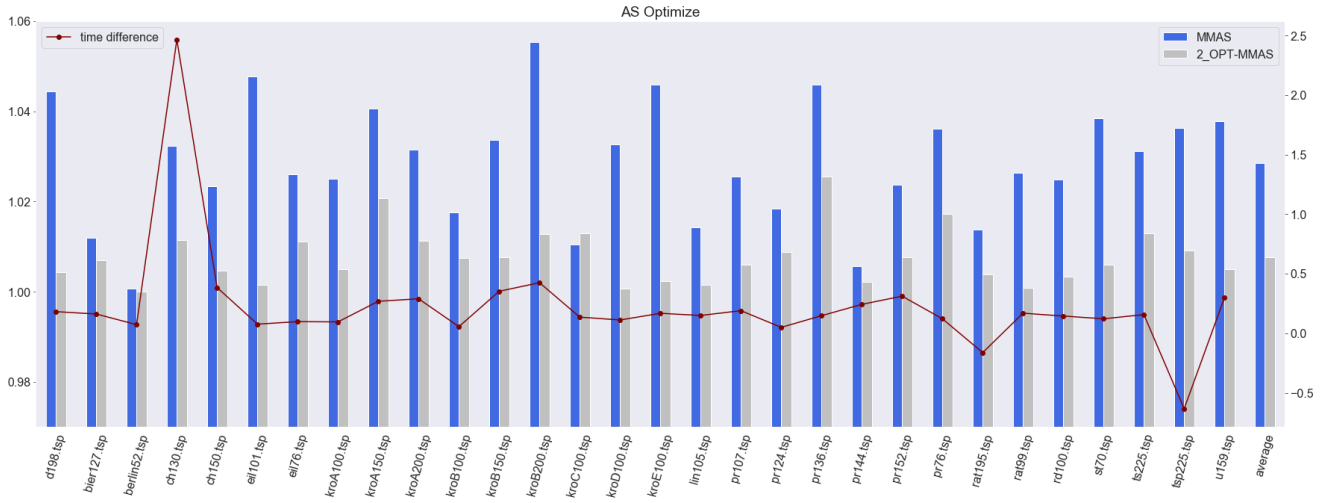


图 5-1-1 蚁群算法优化

5.2 模拟退火算法优化

改进方法是使用贪心策略初始化路径。

如图 5-2-1 所示，改进的效果不是很稳定，在部分实例中甚至出现了负优化，这是因为一个好的初始路径会使得算法陷入局部最优的可能性变大。但是从平均结果看，使用贪心算法初始路径对于模拟退火算法的结果还是有所优化的。

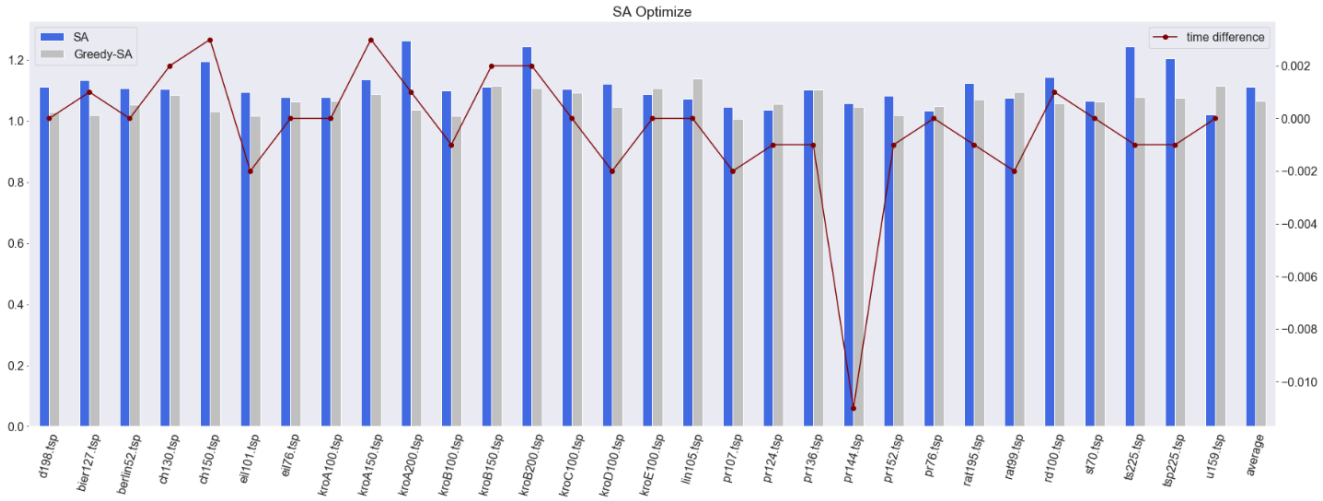


图 5-2-1 模拟退火算法优化

5.3 禁忌搜索算法优化

改进方法是使用改进后的模拟退火算法的解作为初始的路径。

如图 5-3-1 所示，相比于模拟退火算法的改进，禁忌搜索的改进结果更加稳定，而从平均结果上看，禁忌搜索的改进增大了模拟退火算法的改进幅度。

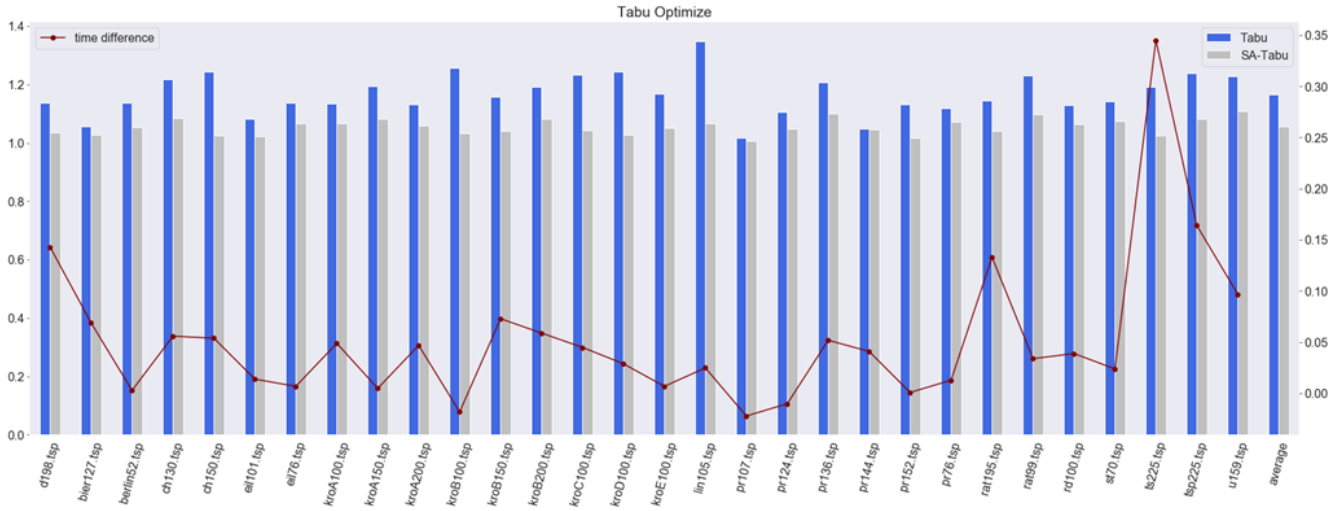


图 5-3-1 禁忌搜索算法优化

6 GUI 与可视化

为了方便查看结果和运行情况，使用 python 设计了图形交互界面进行算法的可视化。算法使用 c++编写，GUI 调用 c++编译生成的可执行文件输出结果文件，进行可视化。
使用图形交互界面和可视化，使得实验结果可读性更强，极大的方便了算法的调参和改进。

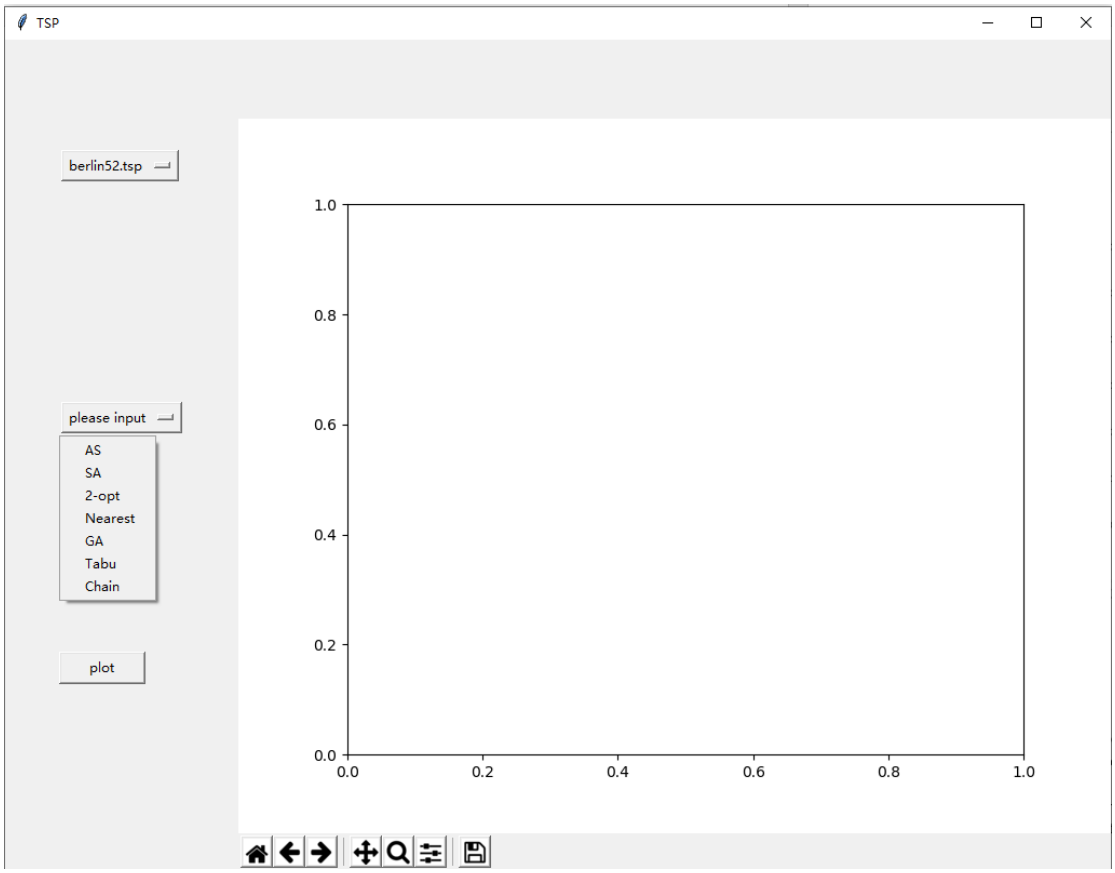


图 6-1 交互界面

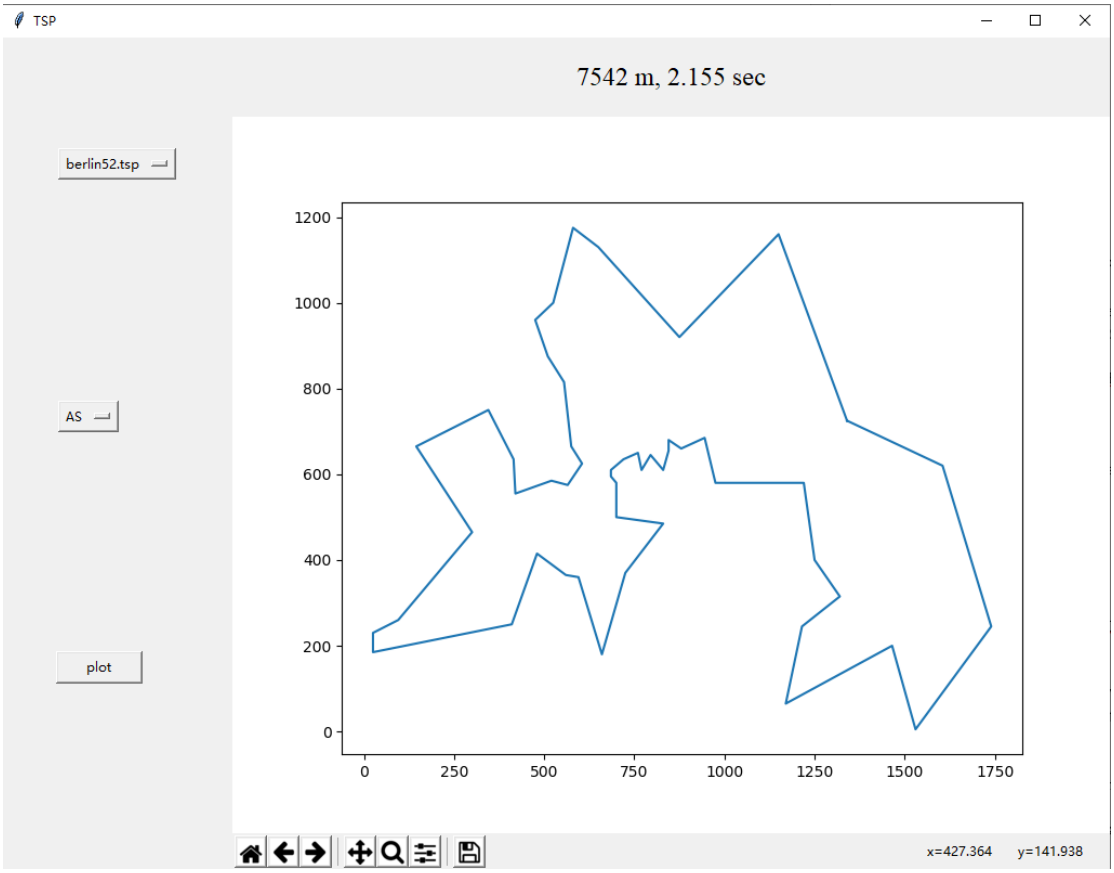


图 6-2 运行结果

7 结论

实现了最近邻算法、2-opt 算法、最短链接算法、遗传算法、禁忌搜索算法、蚁群算法和模拟退火算法，对 benchmark 中的 30 种 TSP 实例以及 gr229(地理距离)进行了实验与分析。使用 GUI 与可视化辅助算法改进，改进后的蚁群算法在求解精确度和稳定性上都有出色的表现，但是时间上不够理想；而模拟退火算法有着速度快且精确度相对较高的优点。

参考文献

[1]Hsu C H , Feng W C . A Power-Aware Run-Time System for High-Performance Computing[C]// Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference. ACM, 2005.

[2] OpenMP Architecture Review Board.OpenMP Application Programming Interface , Version 2.5 [EB/ OL] .[2005-05- 12] .http://www.openmp.org .

[3]Pering T , Burd T , Brodersen R.Dynamic Voltage Scaling and the Design of a Low-Power Microprocessor System[C] // Proc of the PowerDriven Microarchitecture Workshop , 1998.

[4]Freeh V W, Pan Feng , Kappiah N , et al.Exploring the Energy-Time Tradeoff in MPI Programs on a Power-Scalable Cluster[C]//Proc of the 19th Int' l Parallel and Distrib ut ed Processing Sy mp , 2005.

[5]PsilogeorgopoulosM , MunteanuM , ChuangT , etal.Contemporary Techniques for Lower Power Circuit Design , PREST Deli verable D2.1[R] .Technical Report D2.1 , University of Sheffield , 1998.

[6]Scott Kirkpatrick, D. Gelatt Jr, Mario P. Vecchi. Optimization by Simmulated Annealing[J]. ence, 1983, 42(3):671-680.

[7]Dorigo M , Maniezzo V , Colorni A . Ant System: Optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybernetics - Part B[J]. IEEE TRANSACTIONS ON CYBERNETICS, 1996, 26(1):29-41.

[8]Glover F .Tabu Search-Part I[J] .O RSA Journal on Computing , 1989,1(3):190-206.

[9]S. Lin, B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. 1973, 21(2):498-516.

[10]James K , Eberhart R C .A Discrete Binary Version of the Particle Swarm Algorithm [C]//Proc of the IEEE Conf on Systems, Man , and Cybernetics , 1997:4104-4109.

[11]秦东各，王长坤.一种基于 2-opt 算法的混合型蚁群算法[J].工业控制计算机,2018,31(01):98-100.

附录

人员分工

成员	组长	组员	组员	组员
学号	20184411	20184545	20184501	20184594
姓名	眭永熙	岳崇建	王晓菲	鲁风
班级	计 1808	计 1808	计 1808	计 1808
任务分工	蚁群算法、模拟退火算法实现，GUI 与可视化，算法封装设计，测试及分析	遗传算法、禁忌搜索算法实现，测试框架设计	最近邻算法、2-opt 算法实现	最短链接算法实现
报告分工	引言，摘要，算	相关概念和工	算法设计最近邻算	算法设计最短链

	法设计蚁群和模拟退火部分，实验结果与分析，算法改进部分，结论	作，算法设计遗传算法和禁忌搜索部分	法和 2-opt 算法部分	接部分

算法优化失败过程描述

遗传算法优化失败过程

在个体的变异方式上首先使用的是染色体部分逆序的变异方式,后采用两点交换和部分逆序结合的变异方式，但发现路径长度增加、准确率下降，经分析后舍弃了两点交换的变异方式。在交叉方式上，首先采取的是顺序交叉映射，经过对比部分交叉映射性能更加良好，但最终发现在大规模的数据上两者的性能相差不多，所以采取了两种交叉方式相结合的方式。在种群的变异概率上，最初设置的增长率较大，导致种群容易丧失自身的特点，后降低了增长率经对比性能更好。交叉概率由 0.8 提升至 0.9，变异概率由 0.2 降低至 0.1，最终路径的长度减少；进一步提升交叉概率至 0.95，结果没有明显变化；进一步降低变异概率至 0.05，结果变差；最终确定交叉概率 0.95，变异概率 0.1。

禁忌搜索优化失败过程

最初邻域的搜索采取随机邻域的搜索方法，但是发现准确率不高，后改用全局搜索的方法，虽然准确率有所提高，但时间的增长过于夸张，经权衡后保留最初的随机邻域搜索，但增加了随机的概率使得邻域的规模扩大，随机邻域的大小由 0.35 上升至 0.5，保证了准确率和时间。禁忌表长由邻域的一半降低至邻域的四分之一，更容易陷入局部最优，结果性能变差，提升至四分之三准确率大幅上升，但时间开销过大，最终恢复二分之一。

模拟退火算法优化失败过程

一开始算法优化主要是优化产生新解部分,原算法采用的随机选择两点将两点间的路径翻转。之后采取启发式信息进行优化的方法，先依据概率选择待插入的点，且一个点与它在当前路径上的相邻点之间的距离越大，被选中的概率也越大；再依据概率选择插入位置，且与待插入的点的距离越小，被选中的概率越大。然后将待插入的点插入到该位置。使用这种优化方法最后模拟退火的效果并不好，算法的结果与最近邻算法接近，原因可能是采用这种启发式算法容易退化成贪心策略。

心得体会

王晓菲心得体会

算法设计总是在寻求运行时间和求解质量之间平衡,此次主要选择了两种在运行时间上较好的算法，适用于解决时效性较强的实际问题。TSP 问题启发式算法包括环路构造法和环路改进法。最近邻算法是 TSP 问题中最著名的算法，最近邻算法的输出适宜作为环路改进算法的输入。在简单的环路改进算法中，最著名的是由 Croes 提出的 2-Opt 算法，对于 2-Opt 算法在达到一个局部最优解前进行多少次交换是其中最重要的问题，这个数目可能非常的大，实验中收敛次数较难寻找。此次 2-Opt 初始环路随机生成，不同初始环路对 2-Opt 的解有不同的影响，用最近邻的输出作为 2-Opt 初始环路在交换次数低时效果并不好，原因是局部搜索可以持续快速地对环路进行优化，初始环路应该包含一定数目可用缺陷。本次课设在编写程序时要与组长设计接口统一是一个小难点，锻炼了项目中团队合作的能力。课程设计初期查阅了大量的 TSP 有关问题资料，然后将理论知识进行实践，提高了独立思考和实际动手能力。

鲁风心得体会

最短链接是基于贪心算法，每次在整个图的范围内选择最短边加入到解集和中。贪心算法在对问题进行求解时，总是做出在当前看来是最好的选择，不从整体最优上加以考虑，他所做出的是在某种意义上的局部最优解，不能保证最后结果是最优的，但能保证是比较优秀的，贪心算法的效率高。在 TSP 问题中，不同两个城市间的距离可能是相等的，在进行选择时，总是选择了其中一条，这也是最短链接不能求出最优解的根本性问题。开始我准备在寻找最短边时，对所有符合条件的最短边（加入解集和后不产生回路，没有分枝）进行记录，得到不同的当前解集和，然后再去递归扩充当前子集合，得到最后的解。但是当一个图中所有边长度都相等，或者大部分边长度都相等时，这样处理的时间空间代价是极高的，经过测试后，我选择了放弃一定的精确度换取时间和空间上的更优。在代码实现时，我遇到的一大难点就是判断待加入的边与解集和中的边是否产生回路，我解决的办法是采用并查集，用一个数组实现：int father[N],其中 father[i]表示元素 i 的父节点，而父节点本身也是这个集合内的元素，father[1]=2 就表示元素 1 的父节点是元素 2，以这种父亲关系来表示元素所属的集合。若 father[i]=i，则说明元素 i 是该集合的根节点。然后通过对并查集的初始化、查找、合并解决这一问题。然后就是后期的接口处理部分，我开始选择使用 C 语言描述算法，后期为了便于测试，要求封装成类并统一接口，这让我对原先的代码进行了较大的调整并较好地复习了下 C++的封装思想。总体来说,这次课设让我对 TSP 问题有了更深的理解，也产生了一些自己的想法，并且锻炼了自己的代码能力和团队协作能力。