

### 第三次作业

题目 1. 用蚁群算法求解城市数为  $n = 10$  的 TSP 问题，数据如下：

1	10
2	0.4000 0.4439
3	0.2439 0.1463
4	0.1707 0.2293
5	0.5171 0.9414
6	0.2293 0.7610
7	0.8732 0.6536
8	0.6878 0.5219
9	0.8488 0.3609
10	0.6683 0.2536
11	0.6195 0.2634

解答. 初始化城市数目  $n = 10$ ，蚂蚁数目  $m = 10$ ，超参数取为  $\alpha = 1, \beta = 5, \rho = 0.5, T = 2, Q = 10$ ，记城市  $i, j$  之间的信息素为  $\tau_{ij}$ ，启发式信息  $\eta_{ij} = (\text{dis}_{ij})^{-1}$ ，其中  $\text{dis}_{ij}$  表示城市  $i, j$  的二范数距离，于是每个蚂蚁从城市  $i$  移动到城市  $j$  的概率为

$$p_{ij} = \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{k \in J} (\tau_{ik})^\alpha (\eta_{ik})^\beta}$$

其中  $J$  表示当前蚂蚁还未走到过的城市编号。当所有蚂蚁走完所有城市后，再对信息素进行更新：

$$\tau'_{ij} \leftarrow (1 - \rho)\tau_{ij} + \Delta\tau_{ij}, \quad \Delta\tau_{ij} = \sum_{k=1}^m \Delta\tau_{ij}^k$$

其中  $\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k}, & \text{第 } k \text{ 个蚂蚁当前路径经过了边 } (i, j), \\ 0, & \text{否则.} \end{cases}$ ， $L_k$  表示当前蚂蚁  $k$  走的路径长度。

使用上述超参数经过 2 次迭代即可找到最优解，如下图 1 所示；下标为固定其他参数值，修改其中单一变量时平均收敛到最优解所需的次数：

$\beta$	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00
平均收敛次数	19.96	8.51	5.11	3.52	2.71	2.47	2.00	1.91	1.83	1.73
$\alpha$	0.00	1.00	2.00	3.00	4.00	5.00				
平均收敛次数	3.78	2.78	12.98	22.79	30.20	31.72				
$\rho$	0.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
平均收敛次数	2.67	2.73	2.74	2.65	2.73	2.75	2.73	3.56	4.02	5.45
$Q$	0.00	10.00	20.00	30.00	40.00	50.00	60.00	70.00	80.00	90.00
平均收敛次数	3.89	2.66	3.13	2.89	2.87	3.10	3.34	3.40	3.05	3.38

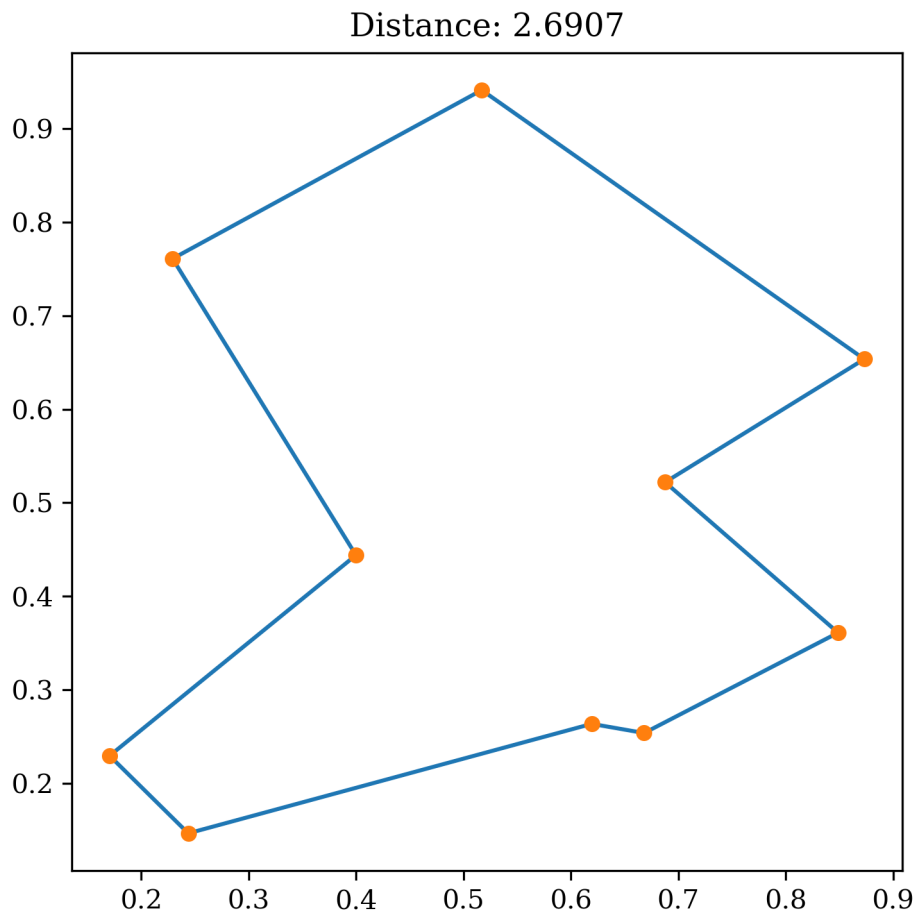


图 1: 最优解

完整代码如下:

```

1  #include <cmath>
2  #include <ctime>
3  #include <vector>
4  #include <cstdio>
5  #include <cstring>
6  #include <cassert>
7  #include <algorithm>
8  #define reset(A, a) std::memset(A, (a), sizeof(A))
9  #define rep(i, n) for (int i = 0; i < n; i++)
10 #define _rep(i, n) for (int i = 1; i <= n; i++)
11
12 const int maxn = 10 + 10; // 城市数上界
13 const int maxm = 10 + 10; // 蚂蚁数上界
14 const int INIT_TAU = 10; // 信息素初值
15 const int maxt = 2; // 迭代重复次数
16 const double ALPHA = 1; // 信息素比重
17 const double BETA = 5; // 启发信息（城市距离）比重
18 const double EPS = 1e-8; // 浮点数精度
19 const double RHO = 0.5; // 信息素残留率
20 const double Q = 10; // 信息素更新时相关常数
21 int n, m = 10;

```

```

22 std::pair<double, double> citys[maxn];
23 double dis[maxn][maxn]; // 邻接表存图
24 void read_data() {
25     freopen("citys.txt", "r", stdin);
26     scanf("%d", &n);
27     rep(i, n) scanf("%lf %lf", &citys[i].first, &citys[i].second);
28     fclose(stdin);
29     rep(i, n) rep(j, n) {
30         double d1 = citys[i].first - citys[j].first, d2 = citys[i].second -
↪ citys[j].second;
31         dis[i][j] = std::sqrt(d1 * d1 + d2 * d2);
32     }
33 }
34 void show_path(int *path) { rep(i, n+1) printf("%d ", path[i]); putchar('\n'); }
35 struct BestState { // 记录最优路径
36     double dis; int path[maxn];
37     BestState() { dis = 1e9; }
38     void update(double d, int *p) {
39         if (d > dis) return;
40         dis = d;
41         rep(i, n+1) path[i] = p[i];
42     }
43     void print() {
44         printf("Minimal distance: %.4lf\nBest path: ", dis);
45         show_path(path);
46     }
47 }best;
48 int pos[maxn]; // 蚂蚁位置
49 int path[maxn][maxn]; // 蚂蚁经过的城市
50 double pathdis[maxn]; // 蚂蚁当前路线长度
51 bool vis[maxn][maxn]; // 蚂蚁访问过的城市
52 double tau[maxn][maxn]; // 每条边上的信息素
53 void debug() {
54     rep(i, n) printf("pathdis[%d]: %lf\n", i, pathdis[i]);
55     best.print();
56     printf("tau:\n");
57     rep(i, n) { rep(j, n) printf("%5.2lf ", tau[i][j]); putchar('\n'); }
58     freopen("best_path.txt", "w", stdout);
59     show_path(best.path);
60     fclose(stdout);
61     freopen("/dev/tty", "w", stdout); // 转到控制台输出
62 }
63 void update() { // 进行一轮信息素更新
64     reset(pathdis, 0); reset(vis, 0);
65     rep(i, m) { // 初始化蚂蚁信息
66         pos[i] = std::rand() % n;
67         path[i][0] = pos[i];
68         vis[i][pos[i]] = 1;
69     }
70     double probab[maxn]; // 临时数组, 计算转移概率
71     for (int t = 1; t <= n; t++) {
72         rep(i, m) {
73             double sum = 0; int next = -1; reset(probab, 0);

```

```

74         if (t == n) next = path[i][0];
75     else {
76         rep(j, n) if (!vis[i][j]) { // 计算转移概率
77             prob[j] = std::pow(tau[pos[i]][j], ALPHA) *
↪ std::pow(1.0/dis[pos[i]][j], BETA);
78             sum += prob[j];
79         }
80         rep(j, n) prob[j] /= sum;
81         double r = (double)std::rand() / RAND_MAX; // 根据随机数求转移城市
82         rep(j, n) if (!vis[i][j]) {
83             if (r <= prob[j] + EPS) { next = j; break; }
84             else r -= prob[j];
85         }
86     }
87     assert(next != -1);
88     pathdis[i] += dis[pos[i]][next];
89     pos[i] = next; path[i][t] = next; vis[i][next] = 1; // 移动
90 }
91 }
92 rep(i, n) rep(j, n) tau[i][j] *= 1 - RHO;
93 rep(i, m) {
94     best.update(pathdis[i], path[i]); // 更新最优解
95     rep(j, n) {
96         int u = path[i][j], v = path[i][j+1];
97         tau[u][v] += Q / pathdis[i]; // 更新每条边上的信息素
98         tau[v][u] += Q / pathdis[i]; // 更新每条边上的信息素
99     }
100 }
101 debug();
102 }
103 void solve() {
104     int T = maxt;
105     rep(i, n) rep(j, n) tau[i][j] = INIT_TAU;
106     while (T--) update();
107 }
108 int main() {
109     srand(2023); // 随机种子
110     read_data();
111     solve();
112     return 0;
113 }

```

---