日期	科目	班级	姓名	学号
2023 年 5 月 31 日	自然计算	强基数学 002	吴天阳	2204210460

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：

```
10
0.4000 0.4439
0.2439 0.1463
0.1707 0.2293
0.5171 0.9414
0.2293 0.7610
0.8732 0.6536
0.6878 0.5219
0.8488 0.3609
0.6683 0.2536
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 τ_{ij}，启发式信息 $\eta_{ij} = (\text{dis}_{ij})^{-1}$，其中 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 $\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, & \text{第}k\text{个蚂蚁当前路径经过了边}(i,j), \\ 0, & \text{否则.} \end{cases}$，$L_k$ 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：

β	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
α	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				
ρ	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

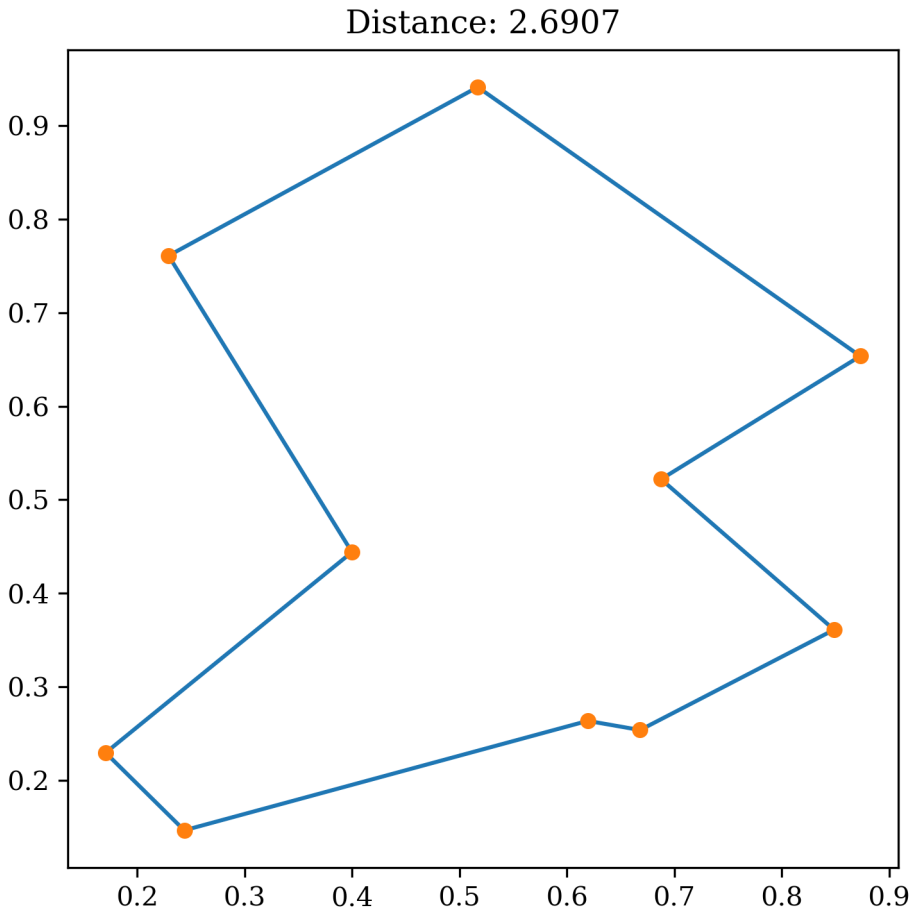

图 1: 最优解

完整代码如下：

```cpp
#include <cmath>
#include <ctime>
#include <vector>
#include <cstdio>
#include <cstring>
#include <cassert>
#include <algorithm>
#define reset(A, a) std::memset(A, (a), sizeof(A))
#define rep(i, n) for (int i = 0; i < n; i++)
#define _rep(i, n) for (int i = 1; i <= n; i++)

const int maxn = 10 + 10;    // 城市数上界
const int maxm = 10 + 10;    // 蚂蚁数上界
const int INIT_TAU = 10;     // 信息素初值
const int maxt = 2;          // 迭代重复次数
const double ALPHA = 1;      // 信息素比重
const double BETA = 5;       // 启发信息（城市距离）比重
const double EPS = 1e-8;     // 浮点数精度
const double RHO = 0.5;      // 信息素残留率
const double Q = 10;         // 信息素更新时相关常数
int n, m = 10;
```

```cpp
std::pair<double, double> citys[maxn];
double dis[maxn][maxn];     // 邻接表存图
void read_data() {
    freopen("citys.txt", "r", stdin);
    scanf("%d", &n);
    rep(i, n) scanf("%lf %lf", &citys[i].first, &citys[i].second);
    fclose(stdin);
    rep(i, n) rep(j, n) {
        double d1 = citys[i].first - citys[j].first, d2 = citys[i].second - citys[j].second;
        dis[i][j] = std::sqrt(d1 * d1 + d2 * d2);
    }
}
void show_path(int *path) { rep(i, n+1) printf("%d ", path[i]); putchar('\n'); }
struct BestState {  // 记录最优路径
    double dis; int path[maxn];
    BestState() { dis = 1e9; }
    void update(double d, int *p) {
        if (d > dis) return;
        dis = d;
        rep(i, n+1) path[i] = p[i];
    }
    void print() {
        printf("Minimal distance: %.4lf\nBest path: ", dis);
        show_path(path);
    }
}best;
int pos[maxm];          // 蚂蚁位置
int path[maxm][maxn];   // 蚂蚁经过的城市
double pathdis[maxm];   // 蚂蚁当前路线长度
bool vis[maxm][maxn];   // 蚂蚁访问过的城市
double tau[maxn][maxn]; // 每条边上的信息素
void debug() {
    rep(i, n) printf("pathdis[%d]: %lf\n", i, pathdis[i]);
    best.print();
    printf("tau:\n");
    rep(i, n) { rep(j, n) printf("%5.2lf ", tau[i][j]); putchar('\n'); }
    freopen("best_path.txt", "w", stdout);
    show_path(best.path);
    fclose(stdout);
    freopen("/dev/tty", "w", stdout);  // 转到控制台输出
}
void update() {  // 进行一轮信息素更新
    reset(pathdis, 0); reset(vis, 0);
    rep(i, m) {  // 初始化蚂蚁信息
        pos[i] = std::rand() % n;
        path[i][0] = pos[i];
        vis[i][pos[i]] = 1;
    }
    double prob[maxn];  // 临时数组，计算转移概率
    for (int t = 1; t <= n; t++) {
        rep(i, m) {
            double sum = 0; int next = -1; reset(prob, 0);
```

```
            if (t == n) next = path[i][0];
            else {
                rep(j, n) if (!vis[i][j]) {  // 计算转移概率
                    prob[j] = std::pow(tau[pos[i]][j], ALPHA) *
↪   std::pow(1.0/dis[pos[i]][j], BETA);
                    sum += prob[j];
                }
                rep(j, n) prob[j] /= sum;
                double r = (double)std::rand() / RAND_MAX;  // 根据随机数求转移城市
                rep(j, n) if (!vis[i][j]) {
                    if (r <= prob[j] + EPS) { next = j; break; }
                    else r -= prob[j];
                }
            }
            assert(next != -1);
            pathdis[i] += dis[pos[i]][next];
            pos[i] = next; path[i][t] = next; vis[i][next] = 1;  // 移动
        }
    }
    rep(i, n) rep(j, n) tau[i][j] *= 1 - RHO;
    rep(i, m) {
        best.update(pathdis[i], path[i]);  // 更新最优解
        rep(j, n) {
            int u = path[i][j], v = path[i][j+1];
            tau[u][v] += Q / pathdis[i];  // 更新每条边上的信息素
            tau[v][u] += Q / pathdis[i];  // 更新每条边上的信息素
        }
    }
    debug();
}
void solve() {
    int T = maxt;
    rep(i, n) rep(j, n) tau[i][j] = INIT_TAU;
    while (T--) update();
}
int main() {
    srand(2023);  // 随机种子
    read_data();
    solve();
    return 0;
}
```