厦門大學

软件学院

《人工智能导论》实验报告

题　　目 模拟退火算法

姓　　名 陈澄

学　　号 32420212202930

班　　级 软工三班

实验时间 2024/04/28

2024 年 **04** 月 **28** 日

1 实验目的

模拟退火算法(Simulated Annealing，SA)最早的思想是由 N. Metropolis 等人于 1953 年提出。1983 年,S. Kirkpatrick 等成功地将退火思想引入到组合优化领域。它是基于 Monte-Carlo 迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。本实验通过解决旅行商问题，帮助学生更好的熟悉和掌握模拟退火算法。

2 实验内容

利用模拟退火算法寻找以下函数的最小值：

f(x)=11*sin(6*x)+7*cos(5*x),x∈[0,2*pi]

3 实验步骤

1. 进行合适的冷却参数表初始化。

参数如下：

```
// 参数设置
double initialTemperature = 1000.0; // 初始温度
double coolingFactor = 0.99; // 温度衰减因子
double stepSize = 0.1; // 搜索步长因子
int iterations = 10000; // 迭代次数
```

2. 编程求得函数最小值

(1)目标函数

```
// 定义目标函数
double objectiveFunction(double x) {
    return 11 * sin(6 * x) + 7 * cos(5 * x);
}
```

(2)新解产生器

随机向正或者负改变一个随机小的改变量，并将新解限制在 a 到 b 之间

```
// 新解产生器：在当前解的基础上进行微小变化
double generateNewSolution(double currentSolution, double stepSize, double a, double b) {
    // 这里以随机生成一个小的浮点数作为变化量
    double change = ((double)rand() / RAND_MAX) * stepSize;
    // 随机决定变化的方向是正还是负
    if (rand() % 2 == 0) {
        // 将新解限制在区间 [a, b] 内
        return (currentSolution + change > b) ? b : currentSolution + change;
    }
    else {
        // 将新解限制在区间 [a, b] 内
        return (currentSolution - change < a) ? a : currentSolution - change;
    }
}
```

(3)Metropolis 接受准则判断是否接受新的解

```
// Metropolis接受准则
bool accept(double delta, double temperature) {
    if (delta < 0) {
        return true; // 总是接受更优解
    }
    else {
        double probability = exp(-delta / temperature);
        double r = ((double)rand() / RAND_MAX);
        return r < probability; // 以一定概率接受劣解
    }
}
```

(4)模拟退火算法

```
// 模拟退火算法求解最小值
double simulatedAnnealing(double initialTemperature, double coolingFactor, double stepSize, int iterations, double a, double b) {
    srand(time(NULL));

    double currentSolution = 0; // 初始解状态
    double temperature = initialTemperature; // 初始温度

    for (int i = 0; i < iterations; ++i) {
        // 产生新解
        double newSolution = generateNewSolution(currentSolution, stepSize, a, b);

        // 计算目标函数差
        double delta = objectiveFunction(newSolution) - objectiveFunction(currentSolution);

        // 判断是否接受新解
        if (accept(delta, temperature)) {
            currentSolution = newSolution;
        }

        // 更新温度
        temperature *= coolingFactor;
    }

    return currentSolution;
}
```

(5)主函数 main

```
int main() {
    // 参数设置
    double initialTemperature = 1000.0; // 初始温度
    double coolingFactor = 0.99; // 温度衰减因子
    double stepSize = 0.1; // 搜索步长因子
    int iterations = 10000; // 迭代次数

    // 调用模拟退火算法求解最小值
    double minSolution = simulatedAnnealing(initialTemperature, coolingFactor, stepSize, iterations, 0, 2 * acos(-1.0));

    cout << "最小值: " << objectiveFunction(minSolution) << endl;
    cout << "最优解: " << minSolution << endl;

    return 0;
}
```

(6)运行结果

3. 总结模拟退火算法优劣。

优点：

(1)全局搜索能力强：模拟退火算法能够在搜索空间中进行全局搜索，不容易陷入局部最优解。

(2)能够接受劣解：通过一定概率接受劣解，有助于跳出局部最优解，增加对全局最优解的探索能力。

(3)易于实现和调试：相对于一些复杂的优化算法，模拟退火算法的实现相对简单，而且参数调优相对容易。

缺点：

(1)计算复杂度较高：模拟退火算法需要进行大量的随机搜索和目标函数计算，因此在解空间较大或者目标函数计算复杂的情况下，计算成本较高。

(2)收敛速度慢：相比于一些局部搜索算法，模拟退火算法的收敛速度通常较慢，在迭代次数有限的情况下可能无法达到理想的解。